A Unified Framework for Speculative Decoding with Multiple Drafters as a Bandit

Taehyeon Kim* **Hojung Jung*** **Se-Young Yun**
KAIST AI
{potter32, ghwjd7281, yunseyoung}@kaist.ac.kr

Abstract

Speculative decoding (SD) has emerged as a promising approach to accelerate inference in large language models (LLMs). This method drafts potential future tokens by leveraging a smaller model, while these tokens are concurrently verified by the target LLM, ensuring only outputs aligned with the target LLM's predictions are accepted. However, the inherent limitations of individual drafters, especially when trained on specific tasks or domains, can hinder their effectiveness across diverse applications. In this paper, we introduce a simple yet efficient unified framework, termed *MetaSD*, that incorporates multiple drafters into the speculative decoding process to address this limitation. Our approach employs multi-armed bandit sampling to dynamically allocate computational resources across various drafters, thereby improving overall generation performance. Through extensive experiments, we demonstrate that our unified framework achieves superior results compared to traditional single-drafter approaches.

1 Introduction

Large language models (LLMs) such as GPT-4 [1], Gemini [34], and Llama [66] have revolutionized real-world applications such as search engine [56], coding assistance, and virtual assistants. However, the token-by-token generation process inherent to LLMs often leads to substantial inference times, primarily due to its memory bandwidth bound nature [54, 60]. Speculative decoding (SD) has emerged as a promising avenue to address this challenge [46, 18]. Precisely, SD employs a smaller draft model (i.e., drafter) to predict potential future tokens. These tokens are verified concurrently by the target LLM, ensuring only outputs aligned with the LLM's predictions are accepted. SD significantly accelerates the generation process, enabling faster and more efficient text generation.

Recent advancements in SD have primarily focused on architectural and training improvements to enhance the acceptance rate of drafted tokens [49, 78, 15, 51, 65]. Notably, techniques such as batched inference and tree verification [65, 51, 15] aim to increase the number of accepted tokens by exploring more decoding paths at one step, while training recipes with knowledge distillation [78, 49] seek to better align the drafter's distribution with that of the target model. However, despite their efficacy in certain tasks, these methods often lack the versatility required to comprehensively cover a wide range of tasks [49, 74]. The inherent limitations of relying on a single drafter, with its specific architectural biases and training data, can hinder performance in scenarios with held-out tasks (Detailed motivation is in Section 2.1).

To mitigate the limitations of single-drafter SD, we propose a novel framework that integrates multiple drafters into the process. Our high level idea is to *meta-draft* the optimal drafter among multiple drafters at test-time utilizing the concept of the exploration-exploitation tradeoff [33]. Effectively utilizing multiple drafters in a real-world serving system presents several challenges. For instance, imagine a scenario where you have several drafters, each specialized for a different task like translation, summarization, or question answering. Determining which drafter will perform best for a

*Equal contribution

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

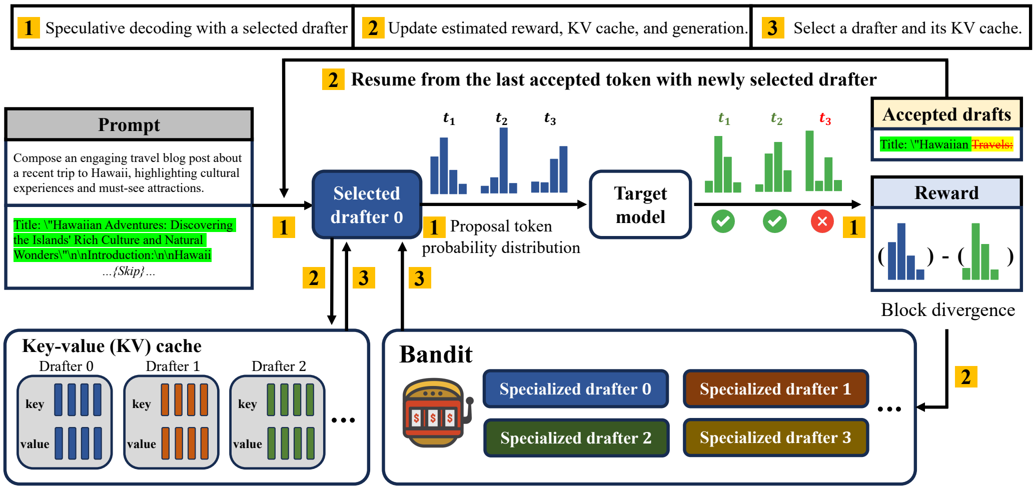
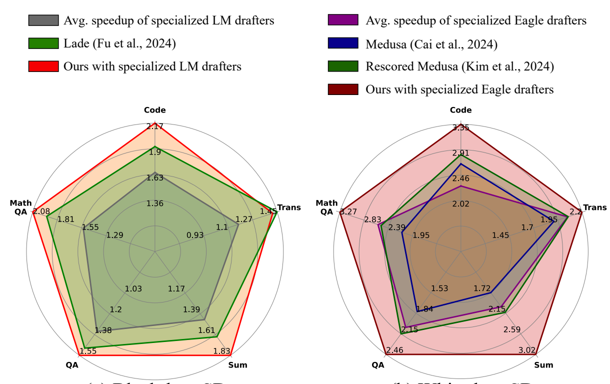


Figure 1: Overview of speculative decoding with multiple drafters in multi-armed bandit (MAB) framework. The example in this figure is from an instance in MT-Bench dataset [77].

given user query is not always straightforward, especially when the query involves multiple tasks or when the topic evolves during the conversation. Furthermore, the system needs to be efficient and adaptable to varying user loads and traffic patterns, without requiring constant manual intervention and parameter tuning. Therefore, an ideal system should have low overhead, meaning it should be robust to variations in user scale or network traffic. It must also be scalable at test time, accurately identifying the optimal drafter for a given query, which is often infeasible in advance, as factors like topic can evolve during inference, making pre-selection unreliable. This dynamic nature of language generation necessitates an adaptive approach.

In the domain of recommendation systems, a similar challenge arises where the optimal set of items to present to a user can change based on their evolving interests and interactions [62]. These systems have successfully employed multi-armed bandit (MAB) algorithms to dynamically adjust recommendations at test time, learning from user feedback to optimize the selection process. Inspired by this approach, we propose a *MetaSD* framework leveraging MAB algorithms to dynamically allocate the optimal drafter among multiple drafters during inference time (Figure 1). This approach enables the system to learn and adapt to the relative performance of each drafter on-the-fly, enabling faster inference. Our key contributions include:



(a) Black-box SD (b) White-box SD

Figure 2: Comparison of average speedup ratios achieved by various SD methods relative to standard autoregressive greedy decoding on a single NVIDIA A100 GPU. The target model is Vicuna 7B v1.3. (a) Results for black-box methods. (b) Results for white-box methods. Detailed description for experimental settings are in **Section 4**.

- We introduce a simple yet efficient framework, termed MetaSD, for incorporating multiple drafters into SD, exploring both black-box approaches where drafters operate independently with access only to the target LLM's predictions and white-box approaches where drafters leverage internal latent features of the target LLM (**Section 2**).
- We establish theoretical upper bounds on the performance of our proposed framework, providing insights into its convergence properties and potential benefits (**Section 3**).
- We demonstrate through extensive experiments that our framework achieves superior inference speed compared to existing single-drafter methods (Figure 2; **Section 4**).

2 Problem statement

2.1 Motivation

Speculative decoding (SD) employs a *draft-verify-accept* paradigm for faster inference. A drafter $\mathcal{M}_q$, which is smaller than the target LLM $\mathcal{M}_p$, drafts the future tokens $\{x^{l+1:l+N_{max}}\}$ based on the input sequence $x^{1:l}$. The target LLM assesses each token x^{l+j} ($j = 1, \ldots, N_{max}$) to determine whether $p(\cdot|x^{1:l+j-1})$ is aligned with its own predictions $q(\cdot|x^{1:l+j-1})$. Only the tokens aligned with the LLM's own predictions are accepted, ensuring the lossless generation (Details in Appendix D).

Despite its advancements, existing works often rely on a single drafter. This reliance can limit the effectiveness of SD, as the drafter's performance is inherently tied to its training data [74, 49]. In scenarios where the drafter's strengths do not align well with the task at hand, its predictions may be less accurate, leading to fewer accepted tokens and diminished speedup benefits of SD. As Table 1 shows, a drafter trained on a specific language pair exhibits significantly higher speedup on that pair compared to others, highlighting the need for a more adaptive approach. Therefore, integrating multiple heterogeneous drafters into the SD framework can potentially address this limitation. By leveraging a pool of drafters, the system can dynamically adapt to varying tasks and input contexts, selecting the most suitable drafter for each situation. From a theoretical and practical viewpoint, the integration of multiple drafters into SD raises several research questions:

Table 1: Speedup ratio relative to the standard autoregressive greedy decoding on various multilingual datasets following [74] where target model is Vicuna 7B v1.3 and the drafter is decoder-only 68M language model: Japanese (Ja)→English (En) [52], Russian (Ru)→En, German (De)→En [14], French (Fr)→En [13], and Chinese (Zh)→En [11]. Evaluations are conducted with a NVIDIA A5000 GPU.

Dataset	Ja-drafter	Ru-drafter	De-drafter	Fr-drafter	Zh-drafter
Ja →En	**1.757** ●	1.109	1.012	1.018	1.154
Ru →En	1.055	**1.817** ●	0.995	0.963	1.036
De →En	1.098	1.369	**2.360** ●	1.036	1.099
Fr →En	1.106	1.445	1.108	**2.135** ●	1.122
Zh →En	1.198	1.086	1.021	1.023	**1.516** ●

1. How to design an efficient and adaptive mechanism for selecting the best drafter at each generation step, considering the exploration-exploitation tradeoff?
2. How to seamlessly incorporate multiple drafters for meta-drafting while minimizing any additional computational overhead?
3. Can we provide theoretical guarantees on the performance of a multi-drafter SD system, ensuring comparable speedup to using single optimal drafter?

To address these challenges, we draw inspiration from the field of multi-armed bandits (MAB). In the MAB framework, an agent repeatedly chooses an action among different choices (arms), each with an unknown reward distribution, aiming to maximize its cumulative reward over time. This closely parallels our problem, where each drafter can be seen as an arm, and the reward is related to the number of accepted tokens or the overall speedup achieved (Algorithm 1). MAB's inherent efficiency and online learning capabilities align well with the requirements of a robust and adaptive multi-drafter SD system. MAB algorithms offer a principled way to balance exploration (trying out different drafters) and exploitation (using the seemingly best drafter) to identify the optimal drafter for each generation step, adapting to the changing context with minimal additional compute costs.

2.2 Problem formulation

Multi-armed bandit (MAB) MAB framework addresses an online learning scenario where, at each round t, an agent takes an action by choosing an arm $a_t \in [K]$ and receives a reward r_t from the environment. The goal of MAB is to design an algorithm that maximizes the expectation of cumulative reward $\mathbb{E}[\sum_{t=1}^{T} r_t]$ throughout a total of T rounds. To achieve this, one can aim to design an optimal policy $\pi^\star$ to minimize the pseudo-regret, defined as: $\textsc{Reg}(\pi, T) = \sum_{t=1}^{T} \mathbb{E}[r_{a_t^\star}] - \mathbb{E}[r_{a_t}]$. Here, a_t denotes the action chosen in round t by the policy π and $a_t^\star$ represents the optimal action in round t which yields the highest expected reward. For a more comprehensive review, we refer the reader to [45].

MetaSD: SD with multiple drafters as a MAB problem We formalize the integration of multiple drafters into SD as a MAB problem, termed as MetaSD framework. Each SD process, consisting

Algorithm 1: MetaSD

INPUT : Drafter pool $[K]$, target model, initial prompt sequence $x^{1:l}$, target sequence length B.
1: $t \leftarrow 0$
2: **while** $l < B$ **do**
3: Meta-draft the drafter i in drafter pool $[K]$ following the bandit
4: Execute one SD step with drafter i and target model given $x^{1:l}$
5: Compute the block divergence between drafter i's predictions and target model's predictions as the reward (Section 2.3)
6: Update the sequence length with the number of accepted tokens from the draft $N_{acc}(i, t)$: $l, t \leftarrow l + N_{acc}(i, t) + 1, t + 1$
7: Update the bandit
8: **end while**

of drafting, verifying, and accepting tokens, corresponds to one round in the MAB setting (Algorithm 1). At round t, a drafter a_t is selected from a pool of heterogeneous drafters $[K]$. The round concludes when all B tokens have been generated. While inspired by classical bandit problems, our MetaSD framework exhibits key distinctions. Unlike classical bandits with a fixed number of rounds, MetaSD operates under a fixed token budget B and the number of total rounds T is stochastic which depends on the policy. Although switching between drafters may incur costs such as prefill cost for tokens and KV cache I/O, we empirically observe that it is negligible in the most of our experiments. Furthermore, for the large scale scenario where switching cost might not be negligible anymore, we provide a detailed discussion with a practical algorithm in Section H.2. While the generated tokens can follow a non-stationary distribution, we assume stationarity within a single turn between the user and the LLM for theoretical analysis. This assumption is reasonable as it allows our framework to be applied with re-intialization for each new query, even in a multi-turn conversation, effectively handling the potential non-stationarity across different queries. In the experiments, MetaSD is implemented with re-initialization for every query.

2.3 Reward design

Ideally, the reward in the MetaSD framework should be informative enough to effectively guide the bandit algorithm towards optimal speedup. One straightforward and readily available choice is the block efficiency (BE), which quantifies the number of mean accepted tokens until a given round [65, 18, 40]. Formally, we define the BE reward for drafter i in round t as: $r_{i,t}^{BE} := N_{acc}(i, t)/N_{max}$, where N_{max} is predefined maximum draft length and $N_{acc}(i, t)$ is number of accepted tokens in the t-th verification stage. While the BE reward provides a direct measure of a drafter's immediate success, it depends on the underlying acceptance rate, denoted as α_i. As shown in [46], this acceptance rate is intrinsically linked to the distance between two probability distributions p and q_i. This implies that by estimating α_i, we can potentially obtain more informative feedback at each round. To leverage this insight, we propose a new reward, coined as block divergence (BD) reward, which estimates the normalized expected number of accepted tokens by utilizing empirical mean of the acceptance rate.

Definition 1 (Block divergence reward)**.** *Let t be the current round, i be the drafter index, and $l(t)$ be the number of input tokens for the target model at round t. Denote $d_{TV}(p^{l(t)}, q_i^{l(t)}) = \frac{1}{2}\|p^{l(t)} - q_i^{l(t)}\|_1$ as the total variation (TV) of two probability measures $p^{l(t)}$ and $q_i^{l(t)}$ from the target model and the drafter i given $x^{1:l(t)}$, respectively. Then, BD reward is defined as follows:*

$$r_{i,t}^{BD} = \frac{1}{N_{max}} \sum_{j=0}^{N_{max}-1} \left(1 - d_{TV}\left(p^{l(t)+j}, q_i^{l(t)+j}\right)\right). \quad (1)$$

While [46] assume a fixed acceptance rate for the j-th candidate in their analysis, we relax this assumption and consider a more general scenario where the acceptance rate for each token follows stationary distribution with mean $\alpha_i \in (0, 1)$ for each drafter $i \in [K]$. Then, one can observe two reward designs are linked by $\mathbb{E}[r_{i,t}^{BE}] = \frac{1-\alpha_i^{N_{max}}}{N_{max}(1-\alpha_i)}\mathbb{E}[r_{i,t}^{BD}]$. As both $\mathbb{E}[r_{i,t}^{BD}]$ and $\mathbb{E}[r_{i,t}^{BE}]$ is monotonically increasing with respect to α_i, maximizing the BD reward aligns with the goal of

Table 2: Reward statistics for BE and BD rewards, collected using autoregressive decoding with the same Japanese dataset and drafter configurations as in Table 1.

Reward statistics	BE reward					BD reward				
	Ja-drafter	Ru-drafter	De-drafter	Fr-drafter	Zh-drafter	Ja-drafter	Ru-drafter	De-drafter	Fr-drafter	Zh-drafter
Ratio of the number of zero rewards	0.503	0.678	0.721	0.743	0.681	-	-	-	-	-
Mean of rewards	0.232	0.099	0.081	0.074	0.106	0.488	0.294	0.317	0.288	0.326
Variance of rewards	0.093	0.032	0.024	0.023	0.037	0.044	0.026	0.032	0.029	0.034

SD, which is to maximize the number of accepted of tokens. We demonstrate that the BD reward empirically and theoretically facilitates the generalization of the MetaSD framework compared to the BE reward, particularly in terms of bandit algorithm performance. To begin, we compare the BD and BE rewards using the following theorem.

Theorem 1 (Informal)**.** *Under the stationary environment, for any reward design r_i with $\mu_i = \mathbb{E}[r_i]$, $i^\star = \arg\max \alpha_i$, and $\Delta_i = \mu_i^\star - \mu_i$, we define the feedback signal for each suboptimal arm $i \neq i^\star$ as*

$$R(r_i) := \frac{\max(\mathrm{Var}[r_i], \mathrm{Var}[r_{i^\star}])}{\Delta_i^2}. \tag{2}$$

Then, for most of the scenarios, $R(r_i^{BD}) < R(r_i^{BE})$.

Theorem 1 demonstrates that the BD reward provides a more informative feedback signal than the BE reward. This signal, defined in eq. 2, plays a crucial role in determining the performance of bandit algorithms. Intuitively, distinguishing two distributions is easier when their expectations are further apart or their variances are smaller. In the context of bandit algorithms, this translates to a smaller regret due to decreased exploration costs. A less noisy feedback signal allows the algorithm to more quickly and accurately identify the optimal arm, reducing the need for extensive exploration of suboptimal arms, as it provides a clearer and more reliable signal for decision-making. Consequently, Theorem 1 implies that we can achieve better performance with bandit algorithms by using the BD reward. In Section G.2, we provide the formal statement of Theorem 1 along with two lemmas providing statistics of the BE reward (Lemma 3) and the BD reward (Lemma 4).

We empirically validate our theoretical analysis regarding the effectiveness of the BD reward compared to the BE reward. For the experiment, we use the same Japanese dataset and drafter configurations as in Table 1, employing autoregressive decoding to collect BE and BD rewards at each step without actual speculative execution. Table 2 reveals striking differences. The BD reward exhibits larger gaps between the expected rewards of the best and suboptimal drafters (Δ_i), while also demonstrating consistently lower variance across all drafters. Consequently, the BD reward has smaller feedback signal R and we can expect using the BD reward leads to more stable learning and faster convergence of the MAB algorithm, enabling faster identification of the optimal drafter. Further explanation is in Section F.6 with Figure 4.

3 Method

This section presents our main method, MetaSD-UCB, which is designed to guarantee the optimal policy for MetaSD. The main challenge arises from the fact that existing regret bounds does not fit into the objective of SD anymore. Moreover, we have to consider stochastic nature of total number of rounds T with the fixed budget B, as opposed to the classical bandit settings where T is fixed. This necessitates us to design a new regret objective and we establish strong regret bounds can still be achieved under this new objective. At the end of this section, we briefly discuss potential extensions, incorporating switching costs between drafters and addressing non-stationary reward distributions.

3.1 Algorithm

MetaSD-UCB We introduce MetaSD-UCB in Algorithm 2, where we combine UCB algorithm [6] in conjunction with the BD reward design to minimize regret. Under the stationary environments, UCB achieves optimal log-linear regret [45]. However, our problem has two key distinctions which prevent direct application of prior analysis. First, the total number rounds required to generate all tokens (i.e., total budget) becomes stochastic. Secondly, minimizing naive regret objective does not guarantee the optimal performance (Section G.3). This arises due to the nature of SD, where the

Algorithm 2: MetaSD-UCB

INPUT Drafter pool $[K]$, initial prompt sequence $x^{1:l}$, target sequence length B, exploration strength hyperparameter β.

$t \leftarrow 0$

/* Phase 1: Meta-draft each drafter in $[K]$ once and do one round of speculative decoding. */

for $i \in [K]$ **do**

  Do one round of SD with drafter i and obtain $N_{acc}(i,t)$, $r_{i,t}$ (by eq. 1)

  $\hat{\mu}_{i,t}, n_i, l, t \leftarrow r_{i,t}, 1, l + N_{acc}(i,t) + 1, t+1$

end for

/* Phase 2: Meta-draft the draft following the UCB bandit until target sequence length B */

while $l < B$ **do**

  $a_t \leftarrow \arg\max_{i \in [K]} \hat{\mu}_{i,t} + \beta\sqrt{\frac{2\ln t}{n_i}}$

  Do one round of SD with drafter a_t and obtain $N_{acc}(a_t,t)$, $r_{a_t,t}$ (by eq. 1)

  $\hat{\mu}_{a_t,t}, n_{a_t}, l, t \leftarrow \frac{\hat{\mu}_{a_t,t} * n_{a_t} + r_{a_t,t}}{n_{a_t}+1}, n_{a_t}+1, l + N_{acc}(a_t,t) + 1, t+1$

end while

performance of the algorithm is determined by total number of rounds until EOS token (or reaching the maximum token length supported by the target LLM). In order to better representing actual speedup, we introduce a novel regret objective for MetaSD, defined as follows.

Definition 2. *Denote $\tau(\pi, B)$ as the number of total rounds of bandit policy π with total budget B and $\pi^\star$ as the optimal policy which satisfies $\pi^\star = \arg\min_\pi \mathbb{E}[\tau(\pi, B)]$. Then, regret objective of MetaSD with policy π becomes:*

$$\textsc{Reg}(\pi, B) = \mathbb{E}\left[\tau(\pi, B)\right] - \mathbb{E}[\tau(\pi^\star, B)]\,. \tag{3}$$

Minimizing eq. 3 is equivalent to maximizing expected number of accepted tokens. This can be seen by observing that the total budget B is consumed by the total number of rounds $\tau(\pi, B)$ plus the total number of accepted tokens across all rounds: $B = \tau(\pi, B) + \sum_{t=1}^{\tau(\pi,B)} N_{acc}(i,t)$. Consequently, minimizing the regret (eq. 3) is directly proportional to maximizing the expected number of accepted tokens, which aligns with the objective of SD.

3.2 Regret upper bound for MetaSD-UCB

We establish that MetaSD-UCB achieves the same level of optimality as the standard UCB [6] by proving that the regret in eq. 3 exhibits a logarithmic growth with respect to the budget B, which is stated in the following theorem.

Theorem 2 (Regret upper bound on MetaSD-UCB)**.** *Denote $\Delta(\alpha_i) = \alpha_{i^\star} - \alpha_i$, where $i^\star$ is the index of the drafter with the largest α_i. Then, using the BD reward, there exists a constant $C > 0$ such that following bound holds:*

$$\textsc{Reg}(\pi, B) < \sum_{i \neq i^\star} \frac{8}{(N_{max})\Delta(\alpha_i)^2} \ln B + C. \tag{4}$$

In Section G.4, we prove the log-linear regret upper bound holds with general reward design but with the higher constant factor $8/\Delta(\alpha_i)^2$. The improvement in eq. 4 stems directly from using the BD reward in Algorithm 2. Since the number of observations within each round grows with N_{max}, the variance of the BD reward is effectively reduced by a factor of N_{max}. This, in turn, leads to a smaller constant term in the regret upper bound compared to using the BE reward. The following corollary captures this observation:

Corollary 1 (Informal)**.** *In most scenarios, the regret upper bound in eq. 4 is tighter than the regret upper bound obtained when using the BE reward with MetaSD-UCB.*

A complete proof of Theorem 2 and a formal statement of Collorary 1 with the proof are in Section G.5.

3.3 Extensions of MetaSD framework

Switching costs In practical implementations, switching between drafters at each round incurs a computational cost due to the need to recalculate previous KV-cache values for the new drafter. This aligns with the concept of bandits with switching costs [10]. However, unlike traditional settings where a fixed cost is incurred per switch, the cost in MetaSD is proportional to the number of unprocessed tokens in the current block. To address this, we propose Algorithm 4 with Sequential Halving (SH) [39], designed specifically for this scenario. A detailed analysis along with theoretical guarantees on its performance is provided in Section H.1.

Non-stationary environment Our prior analysis assumes stationary reward distributions, where the reward feedback for each drafter follows a fixed distribution. However, in certain scenarios, the reward distribution can be non-stationary. For instance, in long-context generation, the optimal drafter might change as the topic or style of the generated text evolves. Despite this challenge, our MetaSD framework remains applicable by leveraging non-stationary bandit algorithms. These algorithms are designed to adapt to changing reward distributions, enabling the system to continuously learn and adjust its drafter selection strategy. Detailed discussions for non-stationary algorithms within the context of MetaSD are in Section H.2.

4 Experiment

4.1 Experimental setup

Models We adopt Vicuna 7B [20] as our target LLM for both black-box and white-box SD. The distinction between two paradigms lies in the drafter's access to the target LLM's internal representations. Black-box drafters operate independently, with access only to the final logit of the target LLM. In contrast, white-box drafters can leverage intermediate activations and hidden states within the target LLM. For black-box SD, we utilize Vicuna 68M [73] as the base architecture for our independent drafters. Each drafter is trained on a distinct task-specific dataset to ensure heterogeneity. Following established practices [41, 78, 15, 74], the training data for these drafters is generated via self-distillation from the target LLM. For white-box SD, we integrate Eagle [47] into the target Vicuna 7B to enable white-box SD. Similar to the black-box setting, multiple Eagle drafters share the same underlying architecture but are fine-tuned on distinct task-specific datasets generated via self-distillation from the target LLM. Further details on the training procedures and datasets used for both black-box and white-box drafters are provided in Appendix F.

Number of drafts N_{max} For black-box SD, we employ speculative sampling (SpS) [18], generating one draft candidate per drafter, termed as MetaSpS. For multi-draft methods like Medusa [15] and Eagle [47], we adhere to their original settings with a tree-attention mechanism. We employ the same tree structure for multiple Eagle drafters described in [47], termed as MetaEagle. Unless explicitly stated otherwise, all approaches utilize a maximum of 5 drafts ($N_{max} = 5$).

Evaluation We conduct evaluations using a NVIDIA A5000, A6000, and A100 GPU under greedy decoding settings. We re-initialize the bandit for each new query, even within multi-turn conversations. Two types of scenarios are evaluated:

1. Diverse task: We evaluate on a diverse range of tasks, including coding (Code) from MT-Bench [77], summarization (Sum) on CNN/Daily [35], De-En translation (Trans) on WMT16 [14], natural question answering (QA) [43], and mathematical reasoning (Math) on GSM8K [21]. The datasets are randomly shuffled to create a non-stationary environment.
2. Multilingual task: We assess the effectiveness in handling multilingual scenarios by evaluating on the multilingual tasks presented in Table 1, following the [74].

The chosen tasks represent a diverse range of applications. Code involves generating text within the constraints of a formal programming language, while Math often requires manipulating symbolic expressions and numerical values. Multilingual tasks introduce challenges related to vocabulary space and token distribution, necessitating drafters tailored to specific language pairs. Summarization highlights the dependency of generation on the input space, where drafters must effectively

Table 3: (Black-box SD) Speedup ratio relative to standard autoregressive greedy decoding on various datasets, comparing single specialized independent drafters, other methods (PLD [58] and Lookahead [28]), and bandit-based drafter selection (Rand (uniformly random), EXP3 [7], SH [39], UCB). Evaluations are conducted with a single NVIDIA A6000 GPU under greedy decoding settings. Drafter specializations: 1: Code, 2: Translation, 3: Summarization, 4: QA, 5: Math.

Speedup	SpS with specialized drafters					Other methods		Bandit in MetaSpS			
	Drafter1	Drafter2	Drafter3	Drafter4	Drafter5	PLD	Lookahead	Rand	EXP3	SH	UCB
Code	**2.437** ●	1.224	1.565	1.814	1.687	1.923	1.542	1.640	1.919	2.148	**2.300** ●
Trans	0.991	**2.076** ●	1.000	1.019	0.950	1.076	1.133	1.150	1.217	1.422	**1.587** ●
Sum	1.513	1.087	**2.133** ●	1.510	1.387	**2.501** ●	1.275	1.429	1.606	1.812	1.971
QA	1.332	1.200	1.343	**1.960** ●	1.252	1.178	1.208	1.294	1.437	1.599	**1.711** ●
Math	1.483	1.228	1.378	1.486	**2.454** ●	1.653	1.533	1.471	1.690	2.144	**2.280** ●

Table 4: (White-box SD) Speedup ratio relative to standard autoregressive greedy decoding on various datasets, comparing single specialized drafters, other methods (blockwise parallel decoding (BPD) [64], Medusa, Rescored-BPD (R-BPD) and Rescored-Medusa [40]), and bandit-based drafter selection. Evaluations are conducted with a single NVIDIA A100 GPU under greedy decoding settings.

Speedup	Specialized Eagle drafters					Other methods				Bandit in MetaEagle			
	Eagle1	Eagle2	Eagle3	Eagle4	Eagle5	BPD	R-BPD	Medusa	R-Medusa	Rand	EXP3	SH	UCB
Code	**3.934** ●	1.303	1.776	2.150	2.427	1.963	2.146	2.661	2.822	2.310	2.858	3.650	**3.724** ●
Trans	1.750	**2.496** ●	2.281	2.131	1.714	1.626	1.442	1.909	2.056	2.036	2.171	2.225	**2.318** ●
Sum	1.707	1.507	**3.382** ●	2.005	1.589	1.509	1.455	1.723	2.136	2.261	2.261	2.801	**3.057** ●
QA	1.842	1.579	2.181	**2.916** ●	1.783	1.489	1.468	1.817	2.154	2.006	2.128	2.466	**2.641** ●
Math	2.584	1.618	2.337	2.433	**3.903** ●	1.696	1.696	2.142	2.519	2.449	2.811	3.339	**3.520** ●

capture and condense information from diverse articles. Finally, QA represents a core natural language understanding task, requiring drafters to comprehend and extract information from complex contexts. For both settings, we utilize a pool of 5 heterogeneous drafters in the MetaSD framework.

4.2 Main result

Diverse task (black-box SD) Table 3 presents the speedup ratios achieved by various methods on a diverse set of tasks using black-box SD. As expected, specialized drafters excel on their respective tasks, as indicated by the highlighted best results. However, their performance suffers significantly on unrelated tasks, demonstrating the limitations of relying on a single drafter. Our MetaSpS-UCB consistently achieves competitive speedup compared to both specialized drafters and other state-of-the-art techniques across all tasks. This highlights the effectiveness of our adaptive drafter selection mechanism in leveraging the strengths of multiple drafters to optimize performance across diverse scenarios. Notably, MetaSpS-UCB reaches the near-optimal performance of the corresponding specialized drafter on several tasks, demonstrating its ability to dynamically identify and utilize the most suitable drafter for the given context. Furthermore, when comparing MetaSpS-UCB to other bandit such as SH and EXP3, considering switching costs and non-stationarity, we observe that MetaSpS-UCB consistently outperforms others. This supports the theoretical advantages of UCB.

Diverse task (white-box SD) Table 4 presents the results for white-box SD with MetaEagle, utilizing EAGLE drafters integrated into the target LLM. Similar to the black-box setting, specialized drafters excel on their designated tasks but struggle on others. MetaEagle-UCB again demonstrates competitive performance, consistently achieving high speedup ratios across all tasks and often outperforming other bandit-based selection strategies. This highlights the adaptability and effectiveness of our proposed framework in both black-box and white-box SD scenarios.

Multilingual task (black-box SD) Table 5 shows the speedup ratios on multilingual tasks. Consistent with the observations in diverse tasks, specialized drafters demonstrate superior performance on their matched language pairs. MetaSps-UCB consistently outperforms other bandit-based selection strategies (EXP3, SH) and remains competitive even with specialized drafters, showcasing its ability to adapt effectively to varying language pairs and achieve notable speedup gains in multilingual scenarios.

Table 5: Speedup ratio relative to standard autoregressive greedy decoding on various multilingual datasets, comparing single specialized drafters to bandit-based drafter selection (EXP3, SH, UCB). Evaluations are conducted with a single NVIDIA A5000 GPU under greedy decoding settings. Drafter specializations: 1: Ja →En, 2: Ru →En, 3: De →En, 4: Fr →En, 5: Zh →En.

Speedup	SpS with specialized drafters					Bandit in MetaSpS		
	Drafter1	Drafter2	Drafter3	Drafter4	Drafter5	EXP3	**SH**	UCB
Ja → En	**1.757** ●	1.109	1.012	1.018	1.154	1.260	**1.368** ●	1.161
Ru → En	1.055	**1.817** ●	0.995	0.963	1.036	1.259	1.403	**1.503** ●
De → En	1.098	1.369	**2.360** ●	1.036	1.099	1.472	1.656	**1.693** ●
Fr → En	1.106	1.445	1.108	**2.135** ●	1.122	1.506	1.607	**1.775** ●
Zh → En	1.198	1.086	1.021	1.023	**1.516** ●	1.204	1.297	**1.369** ●

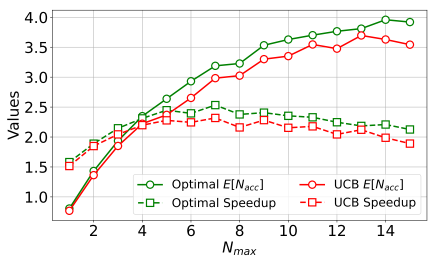


Figure 3: Ablations on N_{max}. 'Optimal' represents the optimal drafter and UCB denotes MetaSps-UCB with BD reward.

Table 6: Average of speedup ratio comparing the BE and BD rewards for MetaSD-UCB with both SpS and EAGLE drafters over 3 different runs.

Task	MetaSpS-UCB		MetaEagle-UCB	
	BE	BD	BE	BD
Code	$2.052_{\pm 0.004}$	$2.231_{\pm 0.006}$ ●	$3.590_{\pm 0.017}$	$3.661_{\pm 0.003}$ ●
Trans	$1.465_{\pm 0.004}$	$1.554_{\pm 0.001}$ ●	$2.228_{\pm 0.009}$ ●	$2.201_{\pm 0.001}$
Sum	$1.770_{\pm 0.002}$	$1.929_{\pm 0.001}$ ●	$3.038_{\pm 0.005}$	$3.043_{\pm 0.001}$ ●
QA	$1.591_{\pm 0.003}$	$1.698_{\pm 0.001}$ ●	$2.629_{\pm 0.003}$ ●	$2.608_{\pm 0.001}$
Math	$1.992_{\pm 0.003}$	$2.238_{\pm 0.002}$ ●	$3.461_{\pm 0.009}$	$3.515_{\pm 0.001}$ ●

Draft length To analyze the impact of draft length on the performance of MetaSps-UCB with the BD reward, we conduct experiments on the Code task using 5 drafters following the same setting in Table 3. The maximum draft length N_{max} is varied to measure the resulting speedup. Figure 3 shows that increasing the draft length initially leads to higher $\mathbb{E}[N_{acc}]$ and speedup due to the increased parallelism in token generation. However, beyond a certain threshold, further increasing the draft length yields diminishing returns and can even decrease performance due to the higher probability of rejection and the associated overhead.

Reward design To assess the impact of our reward function choice, we compare the performance of MetaSD using both BE and BD rewards. In the black-box setting, BD consistently outperforms BE across various tasks, as shown in Table 6. This highlights the importance of utilizing a reward signal that accurately captures the underlying dynamics of the SD process. However, for the MetaEagle-UCB (white-box) setting, both BE and BD rewards exhibit comparable performance. We hypothesize that this is due to Eagle's tree-attention mechanism, which effectively explores multiple decoding paths and implicitly captures the divergence between the drafter and target LLM distributions. This suggests that in white-box settings with multi-path exploration, the choice of reward function might have a less significant impact on the overall performance. Nonetheless, the consistent superiority of BD in the black-box setting underscores its potential benefits in scenarios where such multi-path exploration is not available.

5 Conclusion

In this paper, we introduce a unified framework for incorporating multiple drafters into speculative decoding, addressing the limitations of single-drafter approaches. We formalize this problem as a multi-armed bandit problem, termed as MetaSD, and proposed MetaSD-UCB, a novel algorithm that leverages the Upper Confidence Bound (UCB) principle to dynamically select the optimal drafter at each generation step. We also provide theoretical guarantees on the performance of MetaSD-UCB, establishing its effectiveness in achieving near-optimal speedup even with a stochastic number of rounds. Through extensive experiments on diverse and multilingual tasks, we demonstrate the superior performance of MetaSpS and MetaEagle compared to both specialized drafters and other state-of-the-art methods. Our work opens up new avenues for further research in speculative decoding, including exploring more sophisticated reward designs, incorporating switching costs, and addressing non-stationary environments.

Acknowledgments

This work was partly supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) [No.RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST), 10%, and No. 2022-0-00871, Development of AI Autonomy and Knowledge Enhancement for AI Agent Collaboration, 90%].

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Rajeev Agrawal. Sample mean based index policies by o (log n) regret for the multi-armed bandit problem. *Advances in applied probability*, 27(4):1054–1078, 1995.

[3] Idan Amir, Guy Azov, Tomer Koren, and Roi Livni. Better best of both worlds bounds for bandits with switching costs. *Advances in neural information processing systems*, 35:15800–15810, 2022.

[4] Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. In *COLT-23th Conference on learning theory-2010*, pages 13–p, 2010.

[5] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Tuning bandit algorithms in stochastic environments. In *International conference on algorithmic learning theory*, pages 150–165. Springer, 2007.

[6] P Auer. Finite-time analysis of the multiarmed bandit problem, 2002.

[7] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[8] Peter Auer, Pratik Gajane, and Ronald Ortner. Adaptively tracking the best bandit arm with an unknown number of distribution changes. In *Conference on Learning Theory*, pages 138–158. PMLR, 2019.

[9] Ali Baheri and Cecilia O Alm. Llms-augmented contextual bandit. *arXiv preprint arXiv:2311.02268*, 2023.

[10] Jeffrey S Banks and Rangarajan K Sundaram. Switching costs and the gittins index. *Econometrica: Journal of the Econometric Society*, pages 687–694, 1994.

[11] Loïc Barrault, Ondřej Bojar, Marta R Costa-Jussa, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, et al. Findings of the 2019 conference on machine translation (wmt19). ACL, 2019.

[12] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. *Advances in neural information processing systems*, 27, 2014.

[13] Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58, 2014.

[14] Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. Findings of the 2016 conference on machine translation (wmt16). In *First conference on machine translation*, pages 131–198. Association for Computational Linguistics, 2016.

[15] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

[16] Alexandra Carpentier and Andrea Locatelli. Tight (lower) bounds for the fixed budget best arm identification bandit problem. In *Conference on Learning Theory*, pages 590–604. PMLR, 2016.

[17] Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca, 2023.

[18] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[19] Lijie Chen, Jian Li, and Mingda Qiao. Towards instance optimal bounds for best arm identification. In *Conference on Learning Theory*, pages 535–592. PMLR, 2017.

[20] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023.

[21] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[22] Ofer Dekel, Jian Ding, Tomer Koren, and Yuval Peres. Bandits with switching costs: T 2/3 regret. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 459–467, 2014.

[23] Wenkui Ding, Tao Qin, Xu-Dong Zhang, and Tie-Yan Liu. Multi-armed bandit with budget constraint and variable costs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, pages 232–238, 2013.

[24] Hossein Esfandiari, Amin Karbasi, Abbas Mehrabian, and Vahab Mirrokni. Regret bounds for batched bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35(8), pages 7340–7348, 2021.

[25] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *Computational Learning Theory: 15th Annual Conference on Computational Learning Theory, COLT 2002 Sydney, Australia, July 8–10, 2002 Proceedings 15*, pages 255–270. Springer, 2002.

[26] Eyal Even-Dar, Shie Mannor, Yishay Mansour, and Sridhar Mahadevan. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(6), 2006.

[27] Nicolò Felicioni, Lucas Maystre, Sina Ghiassian, and Kamil Ciosek. On the importance of uncertainty in decision-making with large language models. *arXiv preprint arXiv:2404.02649*, 2024.

[28] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.

[29] Zijun Gao, Yanjun Han, Zhimei Ren, and Zhengqing Zhou. Batched multi-armed bandits problem. *Advances in Neural Information Processing Systems*, 32, 2019.

[30] Aurélien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. In *Conference on Learning Theory*, pages 998–1027. PMLR, 2016.

[31] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *International conference on algorithmic learning theory*, pages 174–188. Springer, 2011.

[32] John Gittins. A dynamic allocation index for the sequential design of experiments. *Progress in statistics*, pages 241–266, 1974.

[33] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[34] Google, Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[35] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

[36] Junya Honda and Akimichi Takemura. An asymptotically optimal bandit algorithm for bounded support models. In *COLT*, pages 67–79. Citeseer, 2010.

[37] Jerry Huang, Prasanna Parthasarathi, Mehdi Rezagholizadeh, and Sarath Chandar. Context-aware assistant selection for improved inference acceleration with large language models. *arXiv preprint arXiv:2408.08470*, 2024.

[38] Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lil'ucb: An optimal exploration algorithm for multi-armed bandits. In *Conference on Learning Theory*, pages 423–439. PMLR, 2014.

[39] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *International conference on machine learning*, pages 1238–1246. PMLR, 2013.

[40] Taehyeon Kim, Ananda Theertha Suresh, Kishore Papineni, Michael Riley, Sanjiv Kumar, and Adrian Benton. Exploring and improving drafts in blockwise parallel decoding, 2024.

[41] Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

[42] Levente Kocsis and Csaba Szepesvári. Discounted ucb. In *2nd PASCAL Challenges Workshop*, volume 2, pages 51–134, 2006.

[43] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[44] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[45] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[46] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

[47] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

[48] Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. Speculative decoding via early-exiting for faster llm inference with thompson sampling control mechanism. *arXiv preprint arXiv:2406.03853*, 2024.

[49] Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Ion Stoica, Zhijie Deng, Alvin Cheung, and Hao Zhang. Online speculative decoding. *arXiv preprint arXiv:2310.07177*, 2023.

[50] Shie Mannor and John N Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5(Jun):623–648, 2004.

[51] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 932–949, 2024.

[52] Makoto Morishita, Katsuki Chousa, Jun Suzuki, and Masaaki Nagata. Jparacrawl v3. 0: A large-scale english-japanese parallel corpus. *arXiv preprint arXiv:2202.12607*, 2022.

[53] Chanwoo Park, Xiangyu Liu, Asuman Ozdaglar, and Kaiqing Zhang. Do llm agents have regret? a case study in online learning and games. *arXiv preprint arXiv:2403.16843*, 2024.

[54] David A Patterson. Latency lags bandwith. *Communications of the ACM*, 47(10):71–75, 2004.

[55] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

[56] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[57] Chloé Rouyer, Yevgeny Seldin, and Nicolo Cesa-Bianchi. An algorithm for stochastic and adversarial bandits with switching costs. In *International Conference on Machine Learning*, pages 9127–9135. PMLR, 2021.

[58] Apoorv Saxena. Prompt lookup decoding, November 2023.

[59] ShareGPT. Sharegpt: Vicuna unfiltered dataset. https://huggingface.co/datasets/Aeala/ShareGPT_Vicuna_unfiltered, 2023. Accessed: 2024.

[60] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[61] Chengshuai Shi, Kun Yang, Zihan Chen, Jundong Li, Jing Yang, and Cong Shen. Efficient prompt optimization through the lens of best arm identification. *arXiv preprint arXiv:2402.09723*, 2024.

[62] Nícollas Silva, Heitor Werneck, Thiago Silva, Adriano CM Pereira, and Leonardo Rocha. Multi-armed bandits in recommendation systems: A survey of the state-of-the-art and future directions. *Expert Systems with Applications*, 197:116669, 2022.

[63] Aleksandrs Slivkins and Eli Upfal. Adapting to a changing environment: the brownian restless bandits. In *COLT*, pages 343–354, 2008.

[64] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

[65] Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. *arXiv preprint arXiv:2310.15141*, 2023.

[66] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[67] Long Tran-Thanh, Archie Chapman, Enrique Munoz De Cote, Alex Rogers, and Nicholas R Jennings. Epsilon–first policies for budget–limited multi-armed bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1211–1216, 2010.

[68] Long Tran-Thanh, Archie Chapman, Alex Rogers, and Nicholas Jennings. Knapsack based optimal policies for budget–limited multi–armed bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1134–1140, 2012.

[69] Peter Whittle. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A):287–298, 1988.

[70] Fanzeng Xia, Hao Liu, Yisong Yue, and Tongxin Li. Beyond numeric awards: In-context dueling bandits with llm agents. *arXiv preprint arXiv:2407.01887*, 2024.

[71] Yu Xia, Fang Kong, Tong Yu, Liya Guo, Ryan A Rossi, Sungchul Kim, and Shuai Li. Which llm to play? convergence-aware online model selection with time-increasing bandits. In *Proceedings of the ACM on Web Conference 2024*, pages 4059–4070, 2024.

[72] Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*, 2023.

[73] Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. Multi-candidate speculative decoding. *arXiv preprint arXiv:2401.06706*, 2024.

[74] Euiin Yi, Taehyeon Kim, Hongseok Jeung, Du-Seong Chang, and Se-Young Yun. Towards fast multilingual llm inference: Speculative decoding and specialized drafters. *arXiv preprint arXiv:2406.16758*, 2024.

[75] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

[76] Yao Zhao, Connor Stephens, Csaba Szepesvári, and Kwang-Sung Jun. Revisiting simple regret: Fast rates for returning a good arm. In *International Conference on Machine Learning*, pages 42110–42158. PMLR, 2023.

[77] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

[78] Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. *arXiv preprint arXiv:2310.08461*, 2023.

Appendices

A Overview of appendix

This appendix provides supplementary material that expands on the main contents. Each section is designed to complement the research presented:

- **Appendix B**: Discusses the broader impact of our work.
- **Appendix C**: Acknowledges the limitations of our current approach and outlines promising directions for future research.
- **Appendix D**: Provides a prepliminary for speculative sampling (SpS).
- **Appendix E**: Provides a comprehensive review of related work, situating our contributions within the broader context of speculative decoding with LLMs and multi-armed bandit research.
- **Appendix F**: Details additional experimental setups, offering further insights into the performance and behavior of our proposed method.
- **Appendix G**: Presents rigorous mathematical proofs for the theoretical guarantees established in the main paper.
- **Appendix H**: Explores extensions to the MetaSD framework, addressing practical considerations such as switching costs and non-stationary environments.
- **Appendix I**: Offers further discussion and analysis of the results presented in the main paper, potentially including additional insights, interpretations, or comparisons.
- **Appendix J**: Provides further experimental results.

Ethics statement This work primarily focuses on improving the efficiency of LLMs through algorithmic advancements and does not directly involve sensitive data or applications that could raise immediate ethical concerns.

Reproducibility statement To facilitate reproducibility, we provide a comprehensive exposition of the materials and experimental configurations within this paper and its accompanying appendices. The organization is as follows:

- **Section 2** - This section presents the problem statement and pseudocode for the MetaSD framework.
- **Section 3** & **Section H.3** - This section provide detailed MAB algorithms for the MetaSD framework under various scenarios.
- **Section 4** - This section elaborates on the implementation specifics, including the pretrained models, datasets, and evaluation metrics.
- **Appendix F** - This section delves into additional details of the experimental settings.

B Broader impact

Generalized speedup Our MetaSD framework for multi-drafter speculative decoding has the potential to enhance the robust speedup capabilities of LLMs. By dynamically selecting from a diverse pool of drafters, the system can better adapt to a wider range of tasks and input contexts, potentially leading to reduced latency on unseen or less frequently encountered scenarios. This increased generalization could benefit various applications, such as machine translation, summarization, and creative writing, where models are often required to handle diverse and unpredictable inputs.

Efficiency The primary goal of our framework is to accelerate the inference process of LLMs. By leveraging speculative decoding with multiple drafters, we aim to achieve significant speedup gains compared to traditional single-drafter approaches. This improved efficiency could enable the deployment of large language models in resource-constrained environments or real-time applications where latency is critical. Faster inference could also facilitate broader accessibility to powerful language models, making them more practical for a wider range of users and use cases.

Systematic impact Our work remains various potential societal impact. Faster and more efficient language models could lead to advancements in various domains, such as healthcare, education, and customer service, where natural language understanding and generation play crucial roles.

C Limitation & Future work

C.1 Limitation

Scalability It is important to acknowledge that the scalability of our approach may be challenged when dealing with an extremely large number of drafters. In such scenarios, the computational overhead associated with evaluating multiple drafters at each step could potentially outweigh the speedup benefits. To address this limitation, future work could explore strategies for pre-selecting a smaller subset of promising drafters based on initial query analysis or other heuristics, before applying the MetaSD framework. This would help to maintain the efficiency and scalability of our approach even in the presence of a vast pool of potential drafters.

Diverse target LLMs While our framework is designed to be agnostic to the target LLM architecture, extensive empirical evaluation across a wider range of LLMs is needed. Future work will assess the generalizability of our approach across different LLM architectures and sizes.

Batched inference Our current implementation primarily focuses on single-query scenarios. Adapting the MetaSD framework to leverage batched inference techniques could further enhance its efficiency, particularly in high-throughput settings.

C.2 Future work

Reward design and exploration-exploitation balance The choice of reward function and the exploration-exploitation tradeoff significantly impact the performance of MetaSD. Exploring alternative reward designs and adaptive exploration strategies could lead to further improvements in speedup and adaptability.

Non-stationarity While we briefly discuss handling non-stationarity in Appendix H, more sophisticated techniques could be investigated. This could involve incorporating change detection mechanisms or developing MAB algorithms specifically tailored to the non-stationary nature of language generation.

Contextual bandits Our current framework primarily relies on observed rewards for drafter selection. Incorporating additional contextual information, such as the query type, user history, or drafter metadata, could lead to more informed decisions. Integrating contextual bandit algorithms into the MetaSD framework is a promising direction for future research.

Reinforcement learning (RL) formulation The MetaSD framework could also be formulated as an RL problem, where the agent learns to select the optimal drafter based on the current state (input context and generated text) to maximize a long-term reward (e.g., overall speedup). Exploring RL-based approaches could potentially uncover novel strategies for adaptive drafter selection.

MAB framework over different SD algorithms Our current work focuses on applying the MAB framework to select among heterogeneous drafters sharing the same SD algorithm (e.g., SpS or EAGLE). While this approach demonstrates significant benefits, it is worth noting that the MAB framework could potentially be extended to encompass a more diverse set of SD algorithms (e.g., Sps, PLD, Lookahead, EAGLE, and others). This would involve designing a reward function and selection strategy that can effectively compare and choose between fundamentally different SD approaches, each with its own strengths and weaknesses. Exploring this broader application of the MAB framework in speculative decoding is an interesting direction for future research.

Algorithm 3: Speculative sampling (SpS)

INPUT : Target LLM $\mathcal{M}_p$, a small drafter $\mathcal{M}_q$, initial prompt sequence $x_1, \ldots, x_l$ and target sequence length B.

while $l < B$ **do**
  for $e \leftarrow 1, \ldots, E$ **do**
    $x_{l_e} \sim \mathcal{M}_q(x|x_1, \ldots, x_l, x_{l_1}, \ldots, x_{l_{e-1}})$
  end for
  In parallel, compute $E + 1$ sets of logits drafts $x_{l_1}, \ldots, x_{l_E}$ with the target LLM $\mathcal{M}_p$:
    $\mathcal{M}_p(x|x_1, \ldots, x_l), \mathcal{M}_p(x|x_1, \ldots, x_l, x_{l_1}), \ldots, \mathcal{M}_p(x|x_1, \ldots, x_l, x_{l_1}, \ldots, x_{l_E})$
  for $j \leftarrow 1, \ldots, E$ **do**
    Sample $r \sim U[0, 1]$ from a uniform distribution
    if $r < \min(1, \frac{\mathcal{M}_p(x|x_1,\ldots,x_{l+j-1})}{\mathcal{M}_q(x|x_1,\ldots,x_{l+j-1})})$ **then**
      Set $x_{l+j} \leftarrow x_{l_j}$ and $l \leftarrow l + 1$
    else
      Sample $x_{l+j} \sim (\mathcal{M}_p(x|x_1, \ldots, x_{l+j-1}) - \mathcal{M}_q(x|x_1, \ldots, x_{l+j-1}))_+$ and exit for loop.
    end if
  end for
  If all tokens $x_{l+1}, \ldots, x_{l+E}$ are accepted, sample extra token $x_{l+E+1} \sim \mathcal{M}_p(x|x_1, \ldots, x_l, x_{l+E})$ and set $l \leftarrow l + 1$
end while

D Preliminary: speculative sampling

Speculative decoding accelerates LLM inference by employing a smaller draft model to predict future tokens, which are then verified by the target LLM. This parallel token generation can significantly reduce latency, especially when the draft model's predictions align well with the target LLM's output distribution.

Algorithm 3 outlines the speculative sampling procedure [46, 18]. Given an initial prompt sequence, the draft model generates E potential future tokens. Concurrently, the target LLM computes the probabilities of these tokens, as well as the probability of its own prediction for each subsequent token position. A drafted token is accepted if its probability, according to the target LLM, exceeds a certain threshold. This threshold is determined by comparing the target LLM's probability for the drafted token to both the draft model's prediction and a random sample, ensuring only high-confidence drafts are accepted. If a drafted token is rejected, the target LLM samples a token from the residual distribution, which represents the difference between its own prediction and the draft model's. This process iterates until the desired sequence length is reached.

Speculative sampling allows the target LLM to process multiple tokens in parallel by drafting them in advance, reducing the overall generation time. When the draft model's predictions are accurate, a significant portion of the generated tokens are accepted, leading to substantial speedup. The verification step and residual sampling ensure that the final generated sequence remains consistent with the target LLM's distribution, preserving generation quality. Speculative sampling provides a foundation for our proposed framework, where we extend this approach to incorporate multiple drafters and dynamically select the optimal one using MAB algorithms.

E Related work

E.1 Speculative decoding

Speculative decoding employs a draft-then-verify paradigm to enhance LLM inference speed. This approach tackles the latency bottleneck in autoregressive decoding, where extensive memory transfers for each token generation lead to underutilized compute resources [54]. Pioneering works by [46, 18] introduced speculative decoding and sampling, enabling lossless acceleration of diverse sampling methods. These methods leverage smaller models within the same model family (e.g., T5-small for T5-XXL) without additional training. Recent advancements have further refined speculative decoding. Models like Eagle [47] and Medusa [15] integrate lightweight feedforward neural network heads into the LLM architecture, enabling early drafting of token sequences and improving throughput.

Despite their efficacy, these methods often rely on a single drafter or a fixed set, limiting adaptability to diverse tasks and input contexts. [74] propose specialized drafters based on the self-distilled dataset training, but dynamically selecting among heterogeneous drafters remains an open challenge. [49] suggest online training of specialized drafters, but their reliance on query-based classification and limited speedup gains highlight the need for a more comprehensive solution.

E.2 Bandit algorithms

Multi-armed bandit Multi-armed bandit (MAB) problem has been extensively studied for decades with various settings. For stochastic MAB setting, [44] and [2] provided asymptotic optimal regret bounds that is logarithmic to the total round T and [6, 5] and [36] proved this result also holds when T is finite. For another variant, EXP3 algorithm [7] proves the optimal regret bound in adversarial environment where reward distribution of each arm can change by adversary in every round.

Budgeted bandit The budgeted MAB problem address a bandit scenario where each arm pull yields both a reward and a cost drawn from individual distributions. Here, the goal is to maximize the cumulative reward until sum of the cost reaches the budget. Then, the optimal arm would be the one with the highest reward-to-cost ratio. ϵ-First policies [67] and KUBE [68] assumed a non-stochastic fixed cost for each arm pull. [23] provided UCB-BV algorithm where cost for each arm is assumed to be a bounded discrete random variable.

Bandits with switching costs In real-world scenarios, a cost may be incurred whenever switching arms. This is related to the MAB problem with switching costs. [22, 29, 57, 24, 3]. For stochastic MAB, [29] and [24] assume a fixed cost is incurred whenever switching arms. They proved an instance-dependent regret bound $O(\log T)$ which does not depend on the unit switching cost value.

Pure exploration Pure exploration or best arm identification (BAI) problems [25, 26, 4] aim to explore as much as possible throughout the round to obtain the best arm at the end of the round. This contrasts with the traditional MAB objective which is maximizing cumulative reward. [25, 50] and [26] investigated pure exploration in MAB under the PAC learning framework. BAI problems are primarily categorized into two settings. First, in the fixed budget setting [4, 39, 16], the goal is to minimize the chance of selecting sub-optimal arms within a fixed number of rounds. The other problem targets fixed confidence setting [39, 38, 30, 19] whose objective is to minimize number of rounds required to achieve a desired confidence level.

Non-stationary bandit Non-stationary bandit problems assume that reward distribution of each arm changes over time. The goal in non-stationary bandit problems is to find a balance between exploration and exploitation while carefully managing past information to adapt to the dynamic environment. Among the earliest works, [32] assumed that only the best arm changes over time. This assumption was later relaxed in [69], where the authors allow the mean reward for each arm to change at every round. [63] assumed reward distribution follows a Brownian motion and established a regret upper bound that grows linear in rounds. Another line of works quantifies the degree of non-stationarity in the bandit instance by assuming a fixed value of L which represents a number of times reward distributions change. [7] suggested EXP3.S algorithm and proved regret upper bound with given L but slightly worse when L is not given. [42] suggested Discounted-UCB, where they obtain reward estimates with discounting factor over time. [31] introduced Sliding-window UCB, where they used fixed-size window to retain information of the rounds within the window for estimating mean reward. ADSWITCH in [8] is proven to be nearly minimax optimal, achieving the state-of-the art regret bound without any prior knowledge of L.

E.3 Large language models and bandits

Recently, several works have made connections between LLMs with bandits using the emergent abilities of LLMs. One side of works utilize LLM as an agent to solve decision making problems combining with bandit framework [9, 27, 70, 53]. On the otherside, some of the works use bandit algorithms for improve the performance guarantee of LLMs with certain tasks such as for efficient prompt optimization [61] and online model selection [71].

Most relevant to ours, several concurrent works investigate how bandit framework can be incorporated into SD. [48] used Thomson sampling algorithm (which is one of the most popular bandit algorithm) to adaptively choose maximum candidate length N_{max} combining with early-exit framework. [37] assumed existence of multiple drafters and formulate SD as a contextual bandit problem. However, they rely on collecting offline samples for the policy learning which can be costly. Furthermore, their approach is regarded as a classification problem that the selected drafter is fixed in a single query. **To the best of our knowledge, our work is the first to use MAB framework within every speculation round and provide its theoretical guarantees.**

F Experiment detail

F.1 Training specialized drafters with self-distilled data

Following the [74], we use their training strategy consisting of two steps:

1. Pretraining drafters on a portion of C4 dataset [55] and ShareGPT dataset [59].
2. Finetuning the models with self distilled data having the target task with templates.

Self-distilled data Following prior work [41, 78, 15, 74], we generate the training data for specialized drafters through self-distillation from the target LLM. To capture the full spectrum of its output variability, we generate multiple responses at various temperatures—$\{0.0, 0.3, 0.7, 1.0\}$. We utilize this self-distilled dataset for training both independent small drafter models and dependent Eagle drafters. For Eagle-specific training details, we adhere to the settings outlined in the original Eagle paper [47].

F.2 Drafter details

All independent drafters are based on a decoder-only Llama transformer model with 68M parameters. The model configuration includes 2 hidden layers, 768 hidden size, 12 attention heads, and a vocabulary size of 32,000. Other key settings are: silu activation function, 0.0 attention dropout, and no weight decay. The training recipe involves pretraining on a subset of the C4 and ShareGPT datasets, followed by fine-tuning on task-specific data generated through self-distillation from the target LLM. We employ 4 NVIDIA A100 GPUs with 80GB memory, utilizing techniques like FSDP (Fully Sharded Data Parallelism), gradient checkpointing, and lazy preprocessing to optimize training efficiency. Hyperparameters include a batch size of 8, 3 training epochs, a learning rate of 2e-5, and a cosine learning rate scheduler with a warmup ratio of 0.03. We maintain consistent architecture and training procedures across all white-box drafters, ensuring their heterogeneity stems solely from the diverse task-specific datasets they are fine-tuned on. For further specifics on Eagle drafter training, we refer readers to the original Eagle paper [47].

F.3 Datasets

Training dataset We utilize a diverse collection of datasets to train our specialized drafters, ensuring their proficiency across various tasks and languages:

- ShareGPT [59]: A dataset of approximately 58,000 conversations scraped. These conversations include both user prompts and responses from OpenAI's ChatGPT.
- WMT16 De→En [14]: A dataset for German-to-English machine translation, providing high-quality parallel text data.
- JparaCrawl-v3.0 [52]: A large-scale Japanese web corpus, enabling training of a drafter specialized in Japanese-to-English translation.
- WMT16 Ru→En [14]: A parallel corpus for Russian-to-English machine translation, similar to the WMT16 De→En dataset but focusing on the Russian language.
- WMT14 Fr→En [13]: A dataset for French-to-English machine translation, providing additional multilingual training data.

- WMT19 Zh→En [11]: A dataset for Chinese-to-English machine translation, further expanding the language coverage of our drafter pool.
- Code alpaca [17]: A dataset of code generation instructions and corresponding outputs, facilitating the training of a drafter specialized in code-related tasks.
- CNN/Daily mail [35]: A dataset for summarization, comprising news articles and their corresponding summaries.
- Natural question answering [43]: A large-scale question answering dataset based on real user queries and Wikipedia passages, aiding in training a drafter for question answering tasks.
- Meta math question answering [75]: A dataset focusing on mathematical question answering, providing specialized training data for a math-oriented drafter.

Evaluation dataset

- Multilingual translation: Ja to En [52], Ru to En, De to En [14], Fr to En [13], and Zh to En [11].
- Code generation: Code tasks from the MT-Bench dataset [77].
- Summarization: CNN/Daily summarization dataset [35].
- Question answering: Natural Questions dataset [43].
- Math reasoning: GSM8K mathematical reasoning dataset [21].

Templates We employ specific prompt templates during model evaluation to guide the behavior of the target LLM and drafters, ensuring consistency and clarity in task execution. These templates are carefully designed to elicit desired responses and provide relevant context for each task category. Before the data templates, system prompts of LLMs are positioned at the front to provide additional context or instructions.

- Multilingual translation: 'Translate this sentence from [source language] to English: [source sentence]'.
- Code generation: Its instruction depends on the query.
- Summarization: 'Summarize: [article text]'.

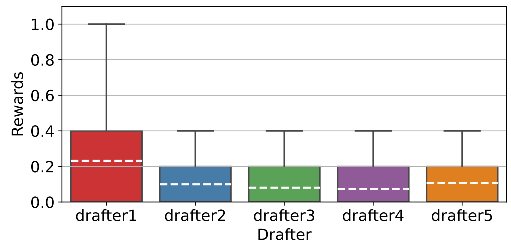
F.4 MAB settings

In our experiments, we set the exploration strength β for MetaSD-UCB to 0.01, balancing exploration and exploitation. For MetaSD-EXP3, we use a gamma value of 0.4 to control the degree of exploration. In the SH algorithm, we set the period to 1, ensuring frequent elimination of underperforming drafters.

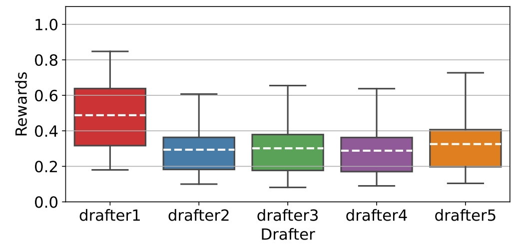
F.5 Baseline

We conduct several SD methods, ensuring their open-source availability and robust performance. Each method embodies a distinct strategy for accelerating LLM inference:

- SpS [18]: SpS employs a smaller LM from the same model series as the drafter. In the verification stage, if a token is rejected, SpS corrects it using residual probability to maintain generation quality.
- BPD, Medusa, and Eagle [64, 15, 47]: These methods enhance the target LLM by incorporating additional lightweight FFN heads. These heads draft potential token sequences based on the penultimate layer representations from the target LLM.
- PLD [58]: Implementing the ideas of [72], PLD selects text spans directly from the input to serve as drafts, aiming for relevant and accurate initial predictions.
- R-BPD (Rescored blockwise parallel decoding) and R-Medusa (Rescored Medusa) [40]: This method enhances BPD by rescoring the drafts at test-time, aiming to increase the number of accepted tokens.



(a) BE reward



(b) BD reward

Figure 4: Comparison of rewards on the Ja→En dataset across different drafters in two scenarios: (a) BE and (b) BD. Box plots show the distribution of rewards, with whiskers extending to the 5th and 95th percentiles. Drafter specializations: 1: Ja →En, 2: Ru →En, 3: De →En, 4: Fr →En, 5: Zh →En.

F.6 Reward distribution

Figure 4 and Table 2 present a statistical analysis of the BE and BD reward distributions, collected using autoregressive decoding with the same Japanese dataset and drafter configurations as in Table 1. Several key observations emerge:

- Lower variance: The BD reward exhibits lower variance compared to the BE reward across all drafters. This suggests that BD provides a more stable and consistent feedback signal, leading to faster convergence with less sample complexity.
- Improved discrimination: The difference in mean reward between the optimal drafter (Drafter 1; Ja-drafter) and the suboptimal drafters is more pronounced with the BD reward. This improved discrimination between drafters can facilitate quicker identification of the optimal drafter by the MAB algorithm.
- Reduced sparsity: A significant portion of the BE rewards are zero, particularly for the suboptimal drafters. This sparsity can hinder the learning process of the MAB algorithm. In contrast, the BD reward consistently provides non-zero feedback, enabling continuous learning and adaptation.

These observations collectively suggest that the BD reward offers several advantages over the BE reward in the context of MetaSD. Its lower variance, improved discrimination between drafters, and reduced sparsity contribute to a more informative and efficient learning signal for the MAB algorithm, potentially leading to faster convergence and better overall performance.

G Proofs

To begin, we provide the mathematical terms and notations in Table 7.

G.1 Basic lemmas

First, we provide basic concentration inequalities which will be used to prove our theoretical results.

Lemma 1 (Chernoff-Hoeffding bound)**.** *Suppose there are n random variables $X_1, X_2, \ldots, X_n$ whose value is bounded in $[0,1]$ and $\mathbb{E}[X_t|X_1,\ldots,X_{t-1}] = \mu$ for $2 \le t \le n$. Then, for $S_n = \sum_{i=1}^n X_i$ and $a \ge 0$, following inequalities holds:*

$$\mathbb{P}(S_n \ge n\mu + a) \le e^{-2a^2/n}, \mathbb{P}(S_n \le n\mu - a) \le e^{-2a^2/n}.$$

Lemma 2 (Bernstein inequality)**.** *Suppose there are n random variables $X_1, X_2, \ldots, X_n$ whose value is bounded in $[0,1]$ and $\sum_{t=1}^n \mathrm{Var}[X_t|X_{t-1},\ldots,X_1] = \sigma^2$. Then, for $S_n = \sum_{i=1}^n X_i$ and $t \ge 0$, following inequalities holds:*

$$\mathbb{P}(S_n \ge \mathbb{E}[S_n] + t) \le \exp(-\frac{t^2}{\sigma^2 + t/2}).$$

G.2 Proof of Theorem 1

In order to prove the theorem, we first provide statistics for the BE and BD rewards by the following lemmas.

BE reward statistics Here, we explicitly calculate expectation and variance of the BE reward in one round of speculative decoding. The result is presented in the following lemma.

Lemma 3 (BE reward statistics)**.** *The expectation and variance of the number of accepted tokens is as follows:*

$$\begin{aligned}\mathbb{E}[r_{i,t}^{BE}] &= \frac{\alpha_i - \alpha_i^{N_{max}+1}}{N_{max}(1-\alpha_i)},\\ \mathrm{Var}[r_{i,t}^{BE}] &= \frac{\alpha_i\left(1-(2N_{max}+1)\alpha_i^{N_{max}} + (2N_{max}+1)\alpha_i^{N_{max}+1} - \alpha_i^{2N_{max}+1}\right)}{(N_{max})^2(1-\alpha_i)^2}.\end{aligned} \tag{5}$$

Proof of Lemma 3 We first start with calculating the expectation and variance of N_{acc} which can be obtained in a closed form. Suppose we conduct one round of speculative decoding for candidate token indices $l+j$ for $j = 1, \ldots, N_{max}$. Now, define E_{l+j}^i as the event of $(l+j)$-th token generated by drafter i is accepted in the verification stage. Also, define random variable X_{l+j}^i to be 1 when E_{l+j}^i occurs and 0 otherwise. With the stationary assumption, one can observe X_{l+j}^i follows Bernoulli distribution with mean α_i. Now, expectation can be obtained as:

$$\mathbb{E}[N_{acc}(i,t)] = \sum_{l=1}^{N_{max}} \mathbb{E}[X_{l+j}^i] = \sum_{l=1}^{N_{max}} \alpha_i^l = \frac{\alpha_i - \alpha_i^{N_{max}+1}}{1-\alpha_i}. \tag{6}$$

To obtain variance, from $X_{L+l}^i \sim Ber(\alpha_i^l)$, following holds:

$$\mathrm{Var}(X_{L+l}^i) = (\alpha_i^l - \alpha_i^{2l})$$

Now, we can directly obtain a closed form of the variance by,

$$
\begin{aligned}
\mathrm{Var}(N_{acc}(i,t)) &= \mathrm{Var}(\sum_{l=1}^{N_{max}} X^i_{L+l}) \\
&= \sum_{l=1}^{N_{max}} \mathrm{Var}(X^i_{L+l}) + 2 \cdot \sum_{l<m} \mathrm{Cov}(X^i_{L+l}, X^i_{L+m}) \\
&= 2 \cdot \sum_{l=1}^{N_{max}} \sum_{m=l}^{N_{max}} \mathrm{Cov}(X^i_{L+l}, X^i_{L+m}) - \sum_{l=1}^{N_{max}} \mathrm{Var}(X^i_{L+l}) \\
&= 2 \cdot \sum_{l=1}^{N_{max}} \sum_{m=l}^{N_{max}} (\alpha_i^m - \alpha_i^{m+l}) - \sum_{l=1}^{N_{max}} (\alpha_i^l - \alpha_i^{2l}) \\
&= 2 \cdot \sum_{l=1}^{N_{max}} \sum_{m=l}^{N_{max}} \alpha_i^m - 2 \cdot \sum_{l=1}^{N_{max}} \sum_{m=l}^{N_{max}} \{\alpha_i^{m+l} - \frac{\alpha_i(1-\alpha_i^{N_{max}})(1-\alpha_i^{N_{max}+1})}{1-\alpha_i^2}\} \\
&= 2 \cdot \sum_{l=1}^{N_{max}} l \cdot \alpha_i^l - 2 \cdot \sum_{l=1}^{N_{max}} \{\alpha_i^l \left(\frac{\alpha_i^l - \alpha_i^{N_{max}+1}}{1-\alpha_i} \right) - \frac{\alpha_i(1-\alpha_i^{N_{max}})(1-\alpha_i^{N_{max}+1})}{1-\alpha_i^2}\} \\
&= 2 \cdot \sum_{l=1}^{N_{max}} l \cdot \alpha_i^l - 2 \cdot \frac{1}{1-\alpha_i} \sum_{l=1}^{N_{max}} \alpha_i^{2l} \\
&+ 2 \cdot \frac{\alpha_i^{N_{max}+1}}{1-\alpha_i} \sum_{l=1}^{N_{max}} \{\alpha_i^l - \frac{\alpha_i(1-\alpha_i^{N_{max}})(1-\alpha_i^{N_{max}+1})}{1-\alpha_i^2}\} \\
&= \frac{2\alpha_i(N_{max} \cdot \alpha_i^{N_{max}+1} - (N_{max}+1)\alpha_i^{N_{max}} + 1)}{(1-\alpha_i)^2} - \frac{2\alpha_i^2(1-\alpha_i^{2N_{max}})}{(1-\alpha_i)(1-\alpha_i^2)} \\
&+ \frac{2\alpha_i^{N_{max}+2}(1-\alpha_i^{N_{max}})}{(1-\alpha_i)^2} - \frac{\alpha_i(1-\alpha_i^{N_{max}})(1-\alpha_i^{N_{max}+1})}{1-\alpha_i^2}.
\end{aligned} \tag{7}
$$

The second equality comes from the basic property of variance, the fourth equality is from observing $\mathrm{Cov}(X^i_{L+l}, X^i_{L+m}) = \mathbb{E}[X^i_{L+l}X^i_{L+m}] - \mathbb{E}[X^i_{L+l}]\mathbb{E}[X^i_{L+m}] = \alpha_i^m - \alpha_i^{l+m}$. After rearranging the terms, we can obtain closed form of the variance as follows.

$$
\mathrm{Var}(N_{acc}(i,t)) = \frac{\alpha_i \left(1 - (2N_{max}+1)\alpha_i^{N_{max}} + (2N_{max}+1)\alpha_i^{N_{max}+1} - \alpha_i^{2N_{max}+1}\right)}{(1-\alpha_i)^2}. \tag{8}
$$

Since $r^{BE}_{i,t} = \frac{1}{N_{max}} N_{acc}(i,t)$ by definition, plugging this into eq. 6 and eq. 8 concludes the proof.

□

BD reward statistics Next, we obtain the expectation and variance of the BD reward by following lemma.

Lemma 4. *Following the relationships hold for $r^{BD}_{i,t}$ for all i, t:*

$$
\mathbb{E}[r^{BD}_{i,t}] = \alpha_i, \mathrm{Var}[r^{BD}_{i,t}] \leq \frac{1}{4N_{max}} \tag{9}
$$

Proof of Lemma 4 Under stationary assumption, any random variable which is bounded in $[0,1]$ has variance less than $\frac{1}{4}$. Since in eq. 1, $r^{BD}_{i,t}$ is constructed by empirical mean of N_{max} numbers of samples under stationary assumption, following holds:

$$
\mathrm{Var}[r_{i,t}] = \mathrm{Var}\left[\frac{1}{N_{max}} \sum_{j=0}^{N_{max}-1} (1 - d_{TV}(p^{l(t)+j}, q_i^{l(t)+j})\right] \leq \frac{1}{4N_{max}},
$$

and this concludes the proof. $\square$

Next, we formally define the bandit signal ratio as follows.

Definition 3 (Feedback signal)**.** *Under stationary environment, any reward design r_i with $\mu_i = \mathbb{E}[r_i]$, $i^\star = \arg\max \mu_i$, and $\Delta_i = \mu_i^\star - \mu_i$, we define feedback signal for each suboptimal arm $i \neq i^\star$ as follows.*

$$R(r_i) := \frac{\max(\mathrm{Var}[r_i], \mathrm{Var}[r_{i^\star}])}{\Delta_i^2}$$

As we will see, R become crucial factor that governs regret upper bound of our MetaSD-UCB algorithm. Specifically, the lower $R(r_i)$ guarantees smaller amount of regret by picking suboptimal arm i.

Then, we provide a formal version of Theorem 1 which states the BD reward actually has lower feedback signal compared to the BE reward.

Theorem 3 (Formal version of Theorem 1)**.** *Denote $\Delta(\alpha_i) := \alpha_{i^\star} - \alpha_i$ for any suboptimal arm i and $n := N_{max}$ for notational convenience. For any $n \in \mathcal{N}$, define functions f_n, g_n, h_n on $(0,1)$ by $f_n(x) = \frac{x - x^{n+1}}{1-x}$, $g_n(x) = f_n'(x) = \sum_{s=1}^n s x^{s-1}$, and $h_n(x) = \sum_{s=1}^n s(x^{s-1} - x^{2n-s})$. Then following holds:*

$$R(r_i^{BD}) \leq \frac{1}{4(\Delta(\alpha_i))^2 N_{max}}. \tag{10}$$

Also, following holds for any drafter configuration satisfying $h_n(\alpha_{i^\star}) \geq \frac{g_n(\alpha_{i^\star})^2}{4n\alpha_{i^\star}}$ and $\mathrm{Var}[r_i^{BE}] < \mathrm{Var}[r_{i^\star}^{BE}]$:

$$R(r_i^{BD}) < R(r_i^{BE}). \tag{11}$$

Proof. Upper bound for the BD reward can be directly obtained from Lemma 4. To prove eq. 11, denote $N_{max} = n$ for notational convenience. Then, by directly applying Lemma 3, it is observed that

$$\begin{aligned}
R(r_i^{BE}) &= \frac{\max(\mathrm{Var}[r_i^{BE}], \mathrm{Var}[r_{i^\star}^{BE}])}{\Delta_i^2} \\
&= \frac{\alpha_{i^\star}(1 - (2n+1)\alpha_{i^\star}^n + (2n+1)\alpha_{i^\star}^{n+1} - \alpha_{i^\star}^{2n+1})}{(f_n(\alpha_i^\star) - f_n(\alpha_i))^2(1-\alpha_{i^\star})^2} \\
&> \frac{\alpha_{i^\star}(1 - (2n+1)\alpha_{i^\star}^n + (2n+1)\alpha_j^{n+1} - \alpha_{i^\star}^{2n+1})}{(g_n(\alpha_{i^\star})\Delta(\alpha_i))^2(1-\alpha_{i^\star})^2} \\
&= \frac{\alpha_{i^\star} h_n(\alpha_{i^\star})}{(g_n(\alpha_{i^\star})\Delta(\alpha_i))^2} \\
&\geq \frac{1}{4(\Delta(\alpha_i))^2 N_{max}},
\end{aligned} \tag{12}$$

where the first inequality is from f_n is a convex function, the second equality comes from Lemma 3, and the last line comes from the assumption.

$\square$

Practical considerations While Theorem 3 provides a general scenario, the inequalities used in its derivation can be quite loose in certain cases. In practice, the BD reward often exhibits a significantly smaller feedback signal $R(r_i)$ than the BE reward. For example, consider the case where $N_{max} = 5$, which is the setting used in our main experiments. The condition $h_n(\alpha_{i^\star}) > \frac{g_n(\alpha_{i^\star})}{4n\alpha_{i^\star}}$ holds for $0.06 < \alpha_{i^\star} < 0.8$, which covers most of the practical range of $\alpha_{i^\star}$. This implies that, in many realistic scenarios, the BD reward leads to a substantially tighter regret bound compared to the BE reward, further supporting its effectiveness in the MetaSD framework. Moreover, assumption of $\mathrm{Var}[r_i^{BE}] < \mathrm{Var}[r_{i^\star}^{BE}]$ covers most of the practical scenarios. As an example, if $n = 5$, $\mathrm{Var}[r_i^{BE}]$ is monotonically increasing until $\alpha_i = 0.815$. Consequently, for any drafter set with $\alpha_{i^\star} < 0.815$, $\mathrm{Var}[r_i^{BE}] < \mathrm{Var}[r_{i^\star}^{BE}]$ holds for all suboptimal drafters.

G.3 Stopping time regret

In this subsection, we provide the equivalence relation between two objectives, maximizing the reward and minimizing the stopping time. First, we define the regret of MetaSD in terms of the stopping time. Denote $\tau(\pi, B)$ as the stopping time for any policy π with budget B and $\pi^\star$ as the optimal policy. In Definition 2, stopping time regret of policy π with budget B is defined as:

$$\textsc{Reg}^s(\pi, B) = \mathbb{E}[\tau(\pi, B)] - \mathbb{E}[\tau(\pi^\star, B)].$$

Intuitively, minimizing $\textsc{Reg}^s(\pi, B)$ should guarantee optimal speedup since minimizing $\tau(\pi, B)$ implies minimizing the number of total SD round. The following lemma proves that our reward design is well aligned with such objective.

Lemma 5 (BE reward original regret)**.** *For any policy π with the given budget B, denote the original regret objective using the BE reward as $\textsc{Reg}^{o,BE}(\pi, T) = \sum_{t=1}^{T}(\mathbb{E}[r_{i^\star}] - \mathbb{E}[r_{a_t}])$. Then, the following equation holds:*

$$\textsc{Reg}^{o,BE}(\pi, T) = \frac{1}{N_{max}}\textsc{Reg}^s(\pi, B)$$

Consequently, minimizing the regret in terms of accepted tokens is equivalent to minimizing $\textsc{Reg}^{(s)}(\pi, B)$.

Proof. It is observed that

$$B = \sum_{t=1}^{\tau(B)}(N_{acc}(i,t) + 1) = \tau(B) + \sum_{t=1}^{\tau(B)} N_{acc}(i,t) = \tau(B) + N_{max}\sum_{t=1}^{\tau(B)} r_{a_t,t}.$$

Thus,

$$\tau(\pi, B) - \tau(\pi^\star, B) = N_{max}\sum_{t=1}^{\tau(B)}(r_{a_t^\star,t} - r_{a_t,t}), \tag{13}$$

where $a_t^\star$ is the action from the optimal policy $\pi^\star$ in round t. By taking the expectation on both sides, we get the result. □

However, we can show that above result does not hold in every reward design.

Lemma 6 (BD reward original regret)**.** *For any policy π with the given budget B, denote the original regret objective using the BE reward as $\textsc{Reg}^{o,BD}(\pi, T) = \sum_{t=1}^{T}(\mathbb{E}[r_{i^\star}] - \mathbb{E}[r_{a_t}])$. Then, there exists a bandit instance with the two different policies π_1, π_2 such that:*

$$\mathbb{E}[Reg^{o,BD}(\pi_1, B)] < \mathbb{E}[Reg^{o,BD}(\pi_2, B)],$$
$$\mathbb{E}[Reg^{s}(\pi_1, B)] > \mathbb{E}[Reg^{s}(\pi_2, B)].$$

Proof. Suppose we have three drafters with $\alpha_1 = 0.1, \alpha_2 = 0.5, \alpha_3 = 0.8$ with $N_{max} = 2$. Consider π_1 as the deterministic policy where it picks the drafter 1 for the first round and pick the drafter 3 rest of the rounds. Also, π_2 be the policy which picks drafter 2 for the first two rounds and drafter 3 for the rest of the rounds. For the original regret objective, π_1 has expected regret of 0.7 while π_2 has expected regret 0.6. However, it can be observed that the number of expected tokens until first two rounds is $(0.1 + 0.1^2) + (0.8 + 0.8^2) = 1.55$ for π_1 and $2(0.5 + 0.5^2) = 1.50$ for π_2. Since policy for the rest of the rounds are the same, we can conclude that the expected stopping time of policy π_1 is less then that of policy π_2. As a result, π_2 is better in terms of original regret objective and π_1 is better with stopping time regret objective. □

G.4 MetaSD-UCB with general reward

In this subsection, we provide a generic theorem which is stated as follows.

Theorem 4 (Generic regret upper bound)**.** *For any reward design r, Denote $\mu_i = \mathbb{E}[r_{i,t}]$, $\Delta_i = \mu_{i^\star} - \mu_i$, and $i^\star = \arg\max \alpha_i$. If $i^\star = \arg\max \mu_i$, then there exists a constant $C'' > 0$ such that following bound holds:*

$$\textsc{Reg}(\pi, B) < \sum_{i \neq i^\star} \frac{8}{\Delta_i^2} \ln B + C''. \tag{14}$$

Above theorem holds for any reward design as long as the drafter with the maximum expected reward $\mathbb{E}[r_{i,t}]$ also has the highest acceptance rate α_i. Since both the BD and BE rewards satisfy this condition, Theorem 4 applies to both of the reward designs. The proof of Theorem 4 consists of two main parts. First, given total round, we can bound the expected number of selecting suboptimal arms using the same anlysis in [6]. Next, we get the upper bound on expected stopping time of MetaSD-UCB algorithm.

Bounding suboptimal selection Given fixed stopping time, we can bound the expectation of number of selecting suboptimal arms as follows:

Lemma 7 (Theorem 1 from [6])**.** *Let $n_i(t)$ be the number of pulling sub-optimal drafter ($i \neq i^\star$) by the MetaSD-UCB until round t. Also, denote $\Delta_i := \mu^r_{i^\star} - \mu^r_i$ be the sub-optimal gap. Then, following inequality holds for $\beta = 1$:*

$$\mathbb{E}[n_i(\tau(B))|\tau(B)] \le \frac{8 \ln \tau(B)}{\Delta_i^2} + 1 + \frac{\pi^2}{3}. \tag{15}$$

Proof of Lemma 7 For the analysis, we restate the proof in [6] for MetaSD-UCB algorithm with our notations. One can observe $n_i(\tau(B))$, the number of times drafter i is chosen for the one round of speculative decoding until the end of generation, can be bounded as follows:

$$\begin{aligned}
&n_i(\tau(B)) = 1 + \sum_{t=K+1}^{\tau(B)} \mathbb{I}[a_t = i] \\
&\le l + \sum_{t=K+1}^{\tau(B)} \mathbb{I}[a_t = i, n_i(t-1) \ge l] \\
&\le l + \sum_{t=K+1}^{\tau(B)} \mathbb{I}\left[\hat{\mu}_{i,t-1} + \sqrt{\frac{2\ln(t-1)}{n_i(t-1)}} \ge \hat{\mu}_{i^\star,t-1} + \sqrt{\frac{2\ln(t-1)}{n_{i^\star}(t-1)}}, n_i(t-1) \ge l\right] \\
&\le l + \sum_{t=1}^{\tau(B)} \sum_{s=1}^{t-1} \sum_{n_i=l}^{t-1} \mathbb{I}\left[\hat{\mu}_{i,n_i} + \sqrt{\frac{2\ln(t-1)}{n_i}} \ge \hat{\mu}_{i^\star,s} + \sqrt{\frac{2\ln(t-1)}{s}}\right].
\end{aligned} \tag{16}$$

Here, $\mathbb{I}$ is an indicator function and l is a positive integer. Now, one can see following holds:

$$\begin{aligned}
&\mathbb{P}\left(\hat{\mu}_{i,n_i} + \sqrt{\frac{2\ln t}{n_i}} \ge \hat{\mu}_{i^\star,s} + \sqrt{\frac{2\ln t}{s}}\right) \le \\
&\mathbb{P}\left(\hat{\mu}_{i^\star,s} \le \mu_{i^\star} - \sqrt{\frac{2\ln t}{s}}\right) + \mathbb{P}\left(\hat{\mu}_{i.n_i} \ge \mu_i + \sqrt{\frac{2\ln t}{n_i}}\right) + \mathbb{P}\left(\mu_{i^\star} < \mu_i + 2 \cdot \sqrt{\frac{2\ln t}{n_i}}\right).
\end{aligned}$$

First term and the second term in the above equation is bounded by Lemma 1 as:

$$\begin{aligned}
&\mathbb{P}\left(\hat{\mu}_{i^\star,s} \le \mu_{i^\star} - \sqrt{\frac{2\ln t}{s}}\right) \le \exp(-4\ln t) = t^{-4}, \\
&\mathbb{P}\left(\hat{\mu}_{i.n_i} \ge \mu_i + \sqrt{\frac{2\ln t}{n_i}}\right) \le \exp(-4\ln t) = t^{-4}.
\end{aligned} \tag{17}$$

By choosing $l = \lceil \frac{8 \ln \tau(B)}{\Delta_i^2} \rceil$, one can see that the last term is 0 since,

$$2 \cdot \sqrt{\frac{2 \ln t}{n_i}} \leq 2 \cdot \sqrt{\frac{2 \ln t}{(\frac{8 \ln \tau(B)}{\Delta_i^2})}} \leq \Delta_i. \tag{18}$$

Finally, taking expectation of eq. 16 and put the above result, one can see that:

$$\begin{aligned} \mathbb{E}[n_i(\tau(B))|\tau(B)] &\leq \lceil \frac{8 \ln \tau(B)}{\Delta_i^2} \rceil + 2 \sum_{t=1}^{\tau(B)} \sum_{s=1}^{t-1} \sum_{n_i=l}^{t-1} 2t^{-4} \\ &\leq \lceil \frac{8 \ln \tau(B)}{\Delta_i^2} \rceil + 2 \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{n_i=l}^{t-1} 2t^{-4} \\ &\leq \frac{8 \ln \tau(B)}{\Delta_i^2} + 1 + \frac{\pi^2}{3}. \end{aligned} \tag{19}$$

□

Bounding stopping time The overall structure of the proof in bounding the stopping time is based on the proof of Lemma 2 in [23] while we provide additional details that suits with our problem formulation. First, we obtain upper bound on stopping time by following lemma:

Lemma 8. *Following inequalities holds for some constants $C' > 0$:*

$$\mathbb{E}[\tau(\pi, B)] \leq \frac{B(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \sum_{i \neq i^\star} \frac{8}{\Delta_i^2} \ln B + C'.$$

In order to prove Lemma 8, we first present two lemmas for bounding stopping time for a single armed bandit process i.e., we play only the single arm consecutively until the end of the round. Then, we provide how can we decouple stopping time of multi-armed bandit process of UCB policy.

Lemma 9. *Let $\tau(\pi^i, B)$ be a stopping time for the single armed bandit process π^i which chooses only same drafter i throughout the generation (i.e. $a_t = i$ for all t). Then the stopping time can be bounded as:*

$$\frac{B(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}} - 1 < \mathbb{E}[\tau(\pi^i, B)] \leq \frac{(B+1)(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}}. \tag{20}$$

Proof. One can see the expected budget consumption in each round is $\mu_i^c = \frac{1-\alpha_i^{N_{max}+1}}{1-\alpha_i}$ and the remaining number of tokens in the last round is contained in $\{1, 2, \cdots, N_{max}\}$. Now, suppose eq. 20 holds for all $B < B_0$. Then one can observe:

$$\begin{aligned} \mathbb{E}[\tau(\pi^i, B_0)] &= \mathbb{E}\left[\sum_{j=0}^{N_{max}} \left(\tau(\pi^i, B_0 - 1 - j) + 1\right) \mathbb{P}[r_i^{BE} = j]\right] \\ &\leq \sum_{j=0}^{N_{max}} \frac{(B_0 - j)(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}} \mathbb{P}[r_i^{BE} = j] + 1 \\ &\leq \sum_{j=0}^{N_{max}} \frac{(B_0 + 1)(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}} \mathbb{P}[r_i^{BE} = j] - \frac{(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}} \mathbb{E}[r_i^{BE} = j] + 1 \\ &= \sum_{j=0}^{N_{max}} \frac{(B_0 + 1)(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}} \mathbb{P}[r_i^{BE} = j]. \end{aligned}$$

Since it is trivial to see that eq. 20 holds for $B = 1$, by mathematical induction, one can conclude the proof. The lower bound can be proved by the exactly same manner as in the upper bound.

□

Now, we propose a lemma which provides an upper bound on expected stopping time.

Lemma 10. *For MetaSD-UCB algorithm π with given token budget B, expectation of stopping time $\tau(B)$ can be bounded as follows:*

$$\mathbb{E}[\tau(B)] \leq \mathbb{E}[\tau(\pi^{i^\star}, B)] + \sum_{i \neq i^\star} \mathbb{E}[n_i(\pi, B)], \tag{21}$$

where, $n_i(\pi, B)$ is number of selecting drafter i by policy π during the generation.

Proof. We first prove the upper bound (eq. 21). For policy π with the budget B, define a corresponding process π^u which is defined by extending the process with the new stopping time, which is:

$$\tau^u(\pi^u, B) = min\{\tau > 0 \mid \sum_{t=1}^{\tau} (N_{acc}(a_t^u, t) + 1) \cdot \mathbb{I}[a_t^u = i^\star] \geq B\}.$$

where, $a_t^u = a_t$ for $t \leq \tau(B)$ and $a_t^u = i^\star$ for $\tau(B) < t \leq \tau^u(\pi^u, B)$. In other words, $\tau^u(\pi^u, B)$ is the time where total number of generated tokens by optimal drafter exceeds B. Then, one can see from the construction of π^u and by observing that τ^u does not depend on the budget consumed by suboptimal arms,

$$\mathbb{E}[n_{i^\star}(\pi, B)] \leq \mathbb{E}[\tau^u(\pi^u, B)] = \mathbb{E}[\tau(\pi^{i^\star}, B)].$$

□

Proof of Lemma 8 To prove the upper bound, from Lemma 7 and Lemma 10, it is shown that

$$\begin{aligned}
\mathbb{E}[\tau(B)] &\leq \mathbb{E}[\tau(\pi^{i^\star}, B)] + \sum_{i \neq i^\star} \mathbb{E}[n_i(\pi, B)] \\
&\leq \frac{(B+1)(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \sum_{i \neq i^\star} \mathbb{E}[n_i(\pi, B)] \\
&\leq \frac{(B+1)(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \sum_{i \neq i^\star} \frac{8}{\Delta_i^2} \mathbb{E}[\ln \tau(B)] + (K-1)(1+\frac{\pi^2}{3}), \\
&\leq \frac{(B+1)\cdot(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \frac{\alpha_{i^\star} - \alpha_{i^\star}^{N_{max}+1}}{1-\alpha_{i^\star}^{N_{max}+1}} \sum_{i \neq i^\star} \frac{8}{\Delta_i^2} \ln \mathbb{E}[\tau(B)].
\end{aligned} \tag{22}$$

where the second inequality holds from Lemma 9, the third inequality holds by Lemma 7, and the last inequality holds from Jensen's inequality. Now, using $\ln(x) \leq \frac{x}{\epsilon} + \ln(\epsilon) - 1$ and taking $\epsilon = \sum_{i \neq i^\star} \frac{16}{\Delta_i^2}$, one can obtain:

$$\mathbb{E}[\tau(B)] \leq \frac{(2B+2)\cdot(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + 2\ln(\sum_{i \neq i^\star} \frac{16}{\Delta_i^2}) - 2 + (2K-2)(1+\frac{\pi^2}{3}).$$

If we again put the above equation into the eq. 22, one can obtain:

$$\begin{aligned}
\mathbb{E}[\tau(B)] &\leq \frac{(B+1)(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \sum_{i \neq i^\star} \frac{8}{\Delta_i^2} \ln\left(\frac{(2B+2)\cdot(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + C_1\right) + C_2 \\
&\leq \frac{B(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \sum_{i \neq i^\star} \frac{8}{\Delta_i^2} \ln B + C',
\end{aligned}$$

where $C_1, C_2, C' > 0$ are appropriate constants independent of B.

□

Proof of Theorem 4 The theorem is proved by observing:

$$\begin{aligned}
\mathbb{E}[\tau(\pi, B)] - \mathbb{E}[\tau(\pi^\star, B)]) &= (\mathbb{E}[\tau(\pi, B)] - \mathbb{E}[\tau(\pi^{i^\star}, B)]) \\
&\le \frac{B(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \sum_{i\neq i^\star} \frac{8}{\Delta_i^2}\ln B + C' - \mathbb{E}[\tau(\pi^{i^\star}, B)]) \\
&< \frac{B(1-\alpha_{i^\star})}{1-\alpha_{i^\star}^{N_{max}+1}} + \sum_{i\neq i^\star} \frac{8}{\Delta_i^2}\ln B + C' - \frac{B(1-\alpha_i)}{1-\alpha_i^{N_{max}+1}} - 1 \\
&< \sum_{i\neq i^\star} \frac{8}{\Delta_i^2}\ln B + C''.
\end{aligned}$$

where $C'' > 0$ is an appropriate constant which doesn't depend on B. First equality comes from Lemma 5, first inequality is from Lemma 8, and second inequality holds by putting $i^\star$ to the lower bound of Lemma 9.

□

Note that above analysis holds for every $\beta > 0$ in Algorithm 2. However, when the budget B is finite, constant terms in the regret bound becomes important which makes the performance of the algorithm dependent on β. We empirically found the optimal β in our experiments.

G.5 Proof of Theorem 2

Concentration inequality Denote empirical mean of the BD and BE rewards as follows.

$$\mu_{i,t}^{BD} = \frac{1}{n_i(t)} \sum_{\tau=1}^{t} r_{i,\tau} \cdot \mathbb{I}[a_\tau = i], \mu_{i,t}^{BE} = \frac{1}{n_i(t)N_{max}} \sum_{\tau=1}^{t} N_{acc}(i,t) \cdot \mathbb{I}[a_\tau = i],$$

where $n_i(t)$ is number of times drafter i is selected until round t and $\mathbb{I}$ is indicator function.

Then, following inequalities can be derived for $\epsilon > 0$:

$$\mathbb{P}\left(\hat{\mu}_i^{BE} \ge \frac{\alpha_i - \alpha_i^{N_{max}+1}}{N_{max}(1-\alpha_i)} + \epsilon\right) \le \exp\left(-\frac{n_i(t)\epsilon^2}{2Var[r_i^{BE}] + \epsilon}\right), \tag{23}$$

$$\mathbb{P}\left(\hat{\mu}_i^{BD} \ge \alpha_i + \epsilon\right) \le \exp\left(-2(N_{max})n_i(t)\epsilon^2\right). \tag{24}$$

eq. 23 comes from combining Bernstein's inequality (Lemma 2) with Lemma 3 and eq. 24 is from combining Hoeffding's inequality (Lemma 1) with Lemma 4.

Bandit algorithm guarantee Using concentration inequalities for both rewards, we provide how the bandit signal defined in eq. 2 directly related to our algorithm Algorithm 2. In the proof of Theorem 4, one can observe that bounding number of suboptimal arm selection (Lemma 7) directly related to the regret under the new regret object defined by stopping time (Definition 2). Leveraging above results, the regret upper bound for MetaSD-UCB algorithm with the BD and BE rewards can be proved.

Proof of Theorem 2 For the BD reward, by putting $\beta = \frac{1}{\sqrt{N_{max}}}$ in the UCB algorithm and apply eq. 24, one can directly observe eq. 17 becomes:

$$\begin{aligned}
&\mathbb{P}\left(\hat{\mu}_{i^\star,s} \le \mu_{i^\star} - \frac{1}{\sqrt{N_{max}}} \cdot \sqrt{\frac{2\ln t}{s}}\right) \le \exp(-4\ln t) = t^{-4}, \\
&\mathbb{P}\left(\hat{\mu}_{i.n_i} \ge \mu_i + \frac{1}{\sqrt{N_{max}}} \cdot \sqrt{\frac{2\ln t}{n_i}}\right) \le \exp(-4\ln t) = t^{-4}.
\end{aligned} \tag{25}$$

By choosing $l = \lceil \frac{8\ln \tau(B)}{(N_{max})\Delta(\alpha_i)^2} \rceil$, one can see for $n_i \ge l$:

$$\frac{2}{\sqrt{N_{max}}} \cdot \sqrt{\frac{2\ln t}{n_i}} \le \Delta_i.$$

Rest of the proof is same as in Theorem 4 and we can obtain:

$$\textsc{Reg}(B) \leq \sum_{i \neq i^\star} \frac{8}{(N_{max})\Delta(\alpha_i)^2} \ln B + C,$$

for some constants $C > 0$ and this concludes the proof of Theorem 2.

□

BE reward regret For MetaSD-UCB algorithm with BE reward, we can obtain regret upper bound by the following theorem.

Theorem 5. *Define $\Delta_i^{BE} := \mu_{i^\star}^{BE} - \mu_i^{BE}$ where $\mu_i^{BE} = \mathbb{E}[r_i^{BE}]$. If $\mathrm{Var}[r_i^{BE}] < \mathrm{Var}[r_{i^\star}^{BE}]$, we can obtain the following regret upper bound for the MetaSD-UCB algorithm using BE reward:*

$$\textsc{Reg}(\pi^{BE}, B) \leq \sum_{i \neq i^\star} \left(\frac{(32\mathrm{Var}[r_{i^\star}^{BE}] + 16)}{(\Delta_i^{BE})^2} \right) \ln B \tag{26}$$

Proof. From eq. 23, one can similarly modify the original proof of the UCB [6].

Then, putting $\epsilon = \sqrt{(8\mathrm{Var}[r_{i^\star}^{BE}] + 4) \ln t}$ into eq. 23 make eq. 17 becomes:

$$\mathbb{P}\left(\hat{\mu}_{i^\star, s} \leq \mu_{i^\star} - \sqrt{(8\mathrm{Var}[r_{i^\star}^{BE}] + 4) \ln t} \right) \leq \exp(-4 \ln t) = t^{-4},$$
$$\mathbb{P}\left(\hat{\mu}_{i.n_i} \geq \mu_i + \sqrt{(8\mathrm{Var}[r_{i^\star}^{BE}] + 4) \ln t} \right) \leq \exp(-4 \ln t) = t^{-4}.$$

By choosing $l = \lceil \frac{(32\mathrm{Var}[r_{i^\star}^{BE}]+16) \ln \tau(B)}{(\Delta_i^{BE})^2} \rceil$, one can see for $n_i \geq l$:

$$2 \cdot \sqrt{\frac{(8\mathrm{Var}[r_{i^\star}^{BE}] + 4) \ln t}{n_i}} \leq \Delta_i^{BE}.$$

Rest of the proof is similar as in Theorem 4. □

Regret comparison We restate the Collorary 1 formally as follows:

Corollary 2. *For any $n \in \mathcal{N}$, define functions f_n, g_n, h_n on $(0, 1)$ by $f_n(x) = \frac{x - x^{n+1}}{1-x}$, $g_n(x) = f_n'(x) = \sum_{s=1}^n s x^{s-1}$, and $h_n(x) = \sum_{s=1}^n s(x^{s-1} - x^{2n-s})$. If $h_n(\alpha_{i^\star}) \geq \frac{g_n(\alpha_{i^\star})^2}{4n\alpha_{i^\star}}$ and $\mathrm{Var}[r_i^{BE}] < \mathrm{Var}[r_{i^\star}^{BE}]$, then the regret of our algorithm π^{BE} with the BE reward feedback is upper bounded by some function $f(B)$, where $f(B) > \frac{8}{(N_{max})(\Delta(\alpha_i))^2} \ln B$.*

Proof. One can observe:

$$\frac{(32\mathrm{Var}[r_i^{BE}] + 16)}{(\Delta_i^{BE})^2} \geq \frac{(32\mathrm{Var}[r_i^{BE}])}{(\Delta_i^{BE})^2} > \frac{16}{\Delta(\alpha_i)^2 (N_{max})},$$

where first inequality comes from Theorem 3. Now, putting above result with Theorem 2 and Theorem 5, we get the result. □

Note that the better regret upper bound does not always guarantee the better performance since sometimes it is a proof artifact. Since we take quite loose inequalities during the proof of Theorem 5, we can improve the constant factors for BE reward. Still, even with assuming we can use Lemma 1 inequality in BE reward (which has better guarantee then Bernstein's inequality), the result of Collorary 1 still holds which shows the distinction between two reward designs in terms of regret as in Theorem 3.

Algorithm 4: Pure exploration-then-commit (PETC)

INPUT Drafter pool $[K]$, initial prompt sequence $x^{1:l}$, target sequence length B, exploration rounds B_0.

1: **for** $l = 1, 2, ..., B_0$ **do**
2: Run SH algorithm with budget B_0 (in Algorithm 5)
3: **end for**
4: $\hat{i}_\star$ be the survived index.
5: **while** $l < B$ **do**
6: SD with a single drafter $\hat{i}_\star$.
7: **end while**

H Extended scenarios for the MetaSD framework

Our MetaSD framework is universal as it can incorporate various bandit algorithms tailored for different scenarios. However, establishing optimality guarantees for existing algorithms in this framework requires careful analysis or one should look for the different algorithm designs. This is due to two key distinctions in our problem formulation: (i) stochastic stopping time, and (ii) a new regret objective defined in terms of this stopping time (Definition 2).

This section explores two distinct scenarios and introduces possible algorithms for each. First, we address a scenario when switching costs is not negligible anymore. In MetaSD framework, this happens when substantial computational or memory overhead is incurred when changing drafters. Second, we consider non-stationary environment where the characteristics of the context change within a one generation. Finally, we briefly discuss on other possible extensions of our framework.

H.1 Switching costs

Switching costs for multiple drafters In order to use multiple drafters in SD, one need to replace all missing key-value(KV) cache values for the model whenever switching one drafter to another. Reading and writing KV cache is one of the factor which can decrease the inference speed, and we define any decrease of inference speed by changing drafter as the switching cost. Formally, switching cost is defined as $\lambda(a_t, t) = \lambda\left(l(t) - l(\tau_i(t))\right) \cdot \mathbb{I}[a_{t-1} \neq a_t]$ where $l(t)$ is number of processed tokens by the target model in round t, $\tau_i(t)$ is the latest round where i-th drafter is selected before round t, $\mathbb{I}$ is an indicator function, and λ is a constant. we first define the pseudo regret objective in the presence of switching costs.

Definition 4. *With bandit policy π and the given budget B, we define the regret as follows:*

$$\text{REG}_{switch}(\pi, B, \lambda) = \mathbb{E}[\tau(\pi, B)] - \mathbb{E}[\tau(\pi^\star, b)] + \sum_{t=2}^{\tau(B)} \lambda_t \mathbb{P}(a_{t-1} \neq a_t). \tag{27}$$

To minimize the above regret, observe $\lambda(\pi, B) = \lambda \sum_{t=1}^{\tau(B)} \lambda(a_t, t) = \lambda \sum_{i=1}^{K} B_i$, where B_i's are total number of tokens generated by the i-th drafter after the final round. Intuitively, this implies that total cost decreases when employing elimination-type of algorithms [4, 39], which successively eliminate sub-optimal drafters and exclude those drafters from future selection. Consequently, the total regret $\text{REG}_{switch}(B, \lambda)$ can be reduced from early elimination of poor-performed drafters. However, regret can still increase if the best drafter is mistakenly eliminated early on. Therefore, it is essential to strike a balance between elimination-based algorithms and standard MAB algorithms. For this, we design a new algorithm **Pure Exploration-Then-Commit** (PETC) in Algorithm 4 which effectively balances these two approaches.

PETC (Algorithm 4) divides the MetaSD into two phases. In the first phase $l < B_0$, the algorithm tries to eliminate sub-optimal drafters as quickly as possible. In the bandit literature, this is related to the pure exploration (or best arm identification) problem [45] and we select using SH Algorithm 5 for our analysis. After the exploration period for estimating the best drafter, the algorithm exclusively selects this drafter for the remaining rounds.

Now, we provide how to find the optimal B_0 which by the following theorem:

Algorithm 5: Sequential Halving (SH) [39]

INPUT Total budget T, drafter pool $[K]$

1: **Initialize** $S_0 \leftarrow [K]$
2: **for** $t = 0, 1, \ldots, \lfloor \log_2(K) \rfloor - 1$ **do**
3: Pull each drafter in S_t for $n_t = \left\lfloor \frac{T}{|S_t| \lceil \log_2(K) \rceil} \right\rfloor$ additional times
4: $R_t(i) \leftarrow \sum_{j=1}^{n_t} r_{i,j}$ for $i \in S_t$
5: Let σ_t be a bijection on S_k such that $R_t(\sigma_t(1)) \leq R_t(\sigma_t(2)) \leq \ldots \leq R_t(\sigma_t(|S_t|))$
6: $S_{k+1} \leftarrow [i \in S_k | R_t(\sigma_t(i)) \leq R_t(\sigma_t(\lceil |S_k|/2 \rceil))]$
7: **end for**

OUTPUT Singleton element of $S_{\lfloor \log_2(K) \rfloor}$

Theorem 6 (Regret upper bound on PETC). *By choosing $B_0 = c \cdot \ln B$ for some constant $c > 0$ and using Algorithm 5 for the pure exploration in the for the first phase in Algorithm 4,* $\text{REG}_{switch}(\pi, B, \lambda) \leq O(\ln B)$ *holds.*

Proof. First, we can decompose the regret as:

$$\text{REG}_{switch}(\pi, B, \lambda) = \sum_{t=1}^{\tau(B_0)} \text{REG}(\pi, t) + \sum_{t=\tau(B_0)+1}^{\tau(B)} \text{REG}(\pi, t) + S_T,$$

where $\text{REG}(\pi, t)$ denotes original regret objective eq. 3 for one round t and S_T denotes the total switching cost. First term can be bounded by the stopping time of selecting the worst drafter every round until B_0 which can be bounded by $\tau(B_0) = O(\ln B)$ according to Lemma 9. To bound the second term, we borrow Theorem 4.1 in [39], where they prove the probability of Sequential Halving algorithm to select the suboptimal arm after B_0 round can be bounded by $3 \log_2 K \cdot \exp(-\frac{B_0}{8H_2 \log_2 K})$, where $H_2 := \max_i \frac{i}{\Delta_i^2}$. Then we have

$$\sum_{t=\tau(B_0)+1}^{\tau(B)} \text{REG}(\pi, t) \leq \tau(\pi^{i_w}, B) \cdot 3 \log_2 K \cdot \exp(-\frac{B_0}{8H_2 \log_2 K}) = O(\ln B),$$

where i_w denotes the worst drafter, $\tau(\pi^{i_w}, B)$ denotes the stopping time for generating B tokens using only the worst drafter. The last term is bounded by $\lambda B_0 = O(\ln B)$ and this concludes the proof. □

Here, we can improve constant term in regret upper bound in Theorem 6 by controlling c according to the switching cost λ and given budget B or we may use more advanced proof techniques in the best arm identification literature such as in [76]. We leave these as a future work.

H.2 Non-stationary environment

In real-world scenarios, the reward distribution for each drafter may evolve over time and past information becomes less relevant for decision-making. This phenomenon, referred to as non-stationarity, challenges traditional MAB algorithms that operate under the assumption of stationary reward distributions. In SD, non-stationarity can stem from various factors. For example, during a long-form text generation task, the optimal drafter may change as the topic or style of the text evolves. Consider the prompt: 'Please summarize and reason about the following article on climate change...'. Initially, a drafter specialized in summarization might be most effective. However, as the generation progresses towards the reasoning part, a drafter trained on logical reasoning tasks could become more suitable.

Non-stationary MetaSD Standard analyses of non-stationary bandits [7, 42, 30] often define L to quantify the number of times the reward distributions change over T rounds. Another line of work [63, 12] quantifies the non-stationarity using V, the total variation of the means. In both cases, the regret (which is often called as dynamic regret) is defined as the cumulative expected difference between the rewards of the optimal arm and the selected arm at each round.

$$\text{REG}(\pi, B, L) = \sum_{t=1}^{\tau(B)} (\max_{i \in [K]} \mu_{i,t} - \mathbb{E}[\mu_{a_t,t}]) \tag{28}$$

Algorithm 6: Discounted UCB in MetaSD

INPUT Drafter pool $[K]$, initial prompt sequence $x^{1:l}$, target sequence length B, exploration strength β, decaying parameter γ.

1: $t \leftarrow 0$
/* Phase 1: Meta-draft each drafter in $[K]$ once and do one round of speculative decoding. */
2: **for** $i \in [K]$ **do**
3: Do one round of SD with drafter i and obtain $N_{acc}(i,t)$, $r_{i,t}$ (by eq. 1)
4: $\hat{\mu}_{i,t}, n_i, l, t \leftarrow r_{i,t}, 1, l + N_{acc}(i,t) + 1, t+1$
5: **end for**
/* Phase 2: Meta-draft the draft following the UCB bandit until target sequence length B */
6: **while** $l < B$ **do**
7: $a_t \leftarrow \arg\max_{i \in [K]} \hat{\mu}_{i,t} + \beta\sqrt{\frac{2\ln t}{n_i}}$
8: Do one round of SD with drafter a_t and obtain $N_{acc}(a_t,t)$, $r_{a_t,t}$ (by eq. 1)
9: $\hat{\mu}_{a_t,t} = \frac{1}{n_{a_t}} \sum_{s=1}^{t} \gamma^{t-s} r_{a_s,s} \mathbb{I}[a_s = a_t]$
10: $n_{a_t}, l, t \leftarrow n_{a_t} + 1, l + N_{acc}(a_t,t) + 1, t+1$
11: **end while**

Algorithm 7: Sliding-window UCB in MetaSD

INPUT Drafter pool $[K]$, initial prompt sequence $x^{1:l}$, target sequence length B, exploration parameter β, window size τ.

1: $t \leftarrow 0$
/* Phase 1: Meta-draft each drafter in $[K]$ once and do one round of speculative decoding. */
2: **for** $i \in [K]$ **do**
3: Do one round of SD with drafter i and obtain $N_{acc}(i,t)$, $r_{i,t}$ (by eq. 1)
4: $\hat{\mu}_{i,t}, n_i, l, t \leftarrow r_{i,t}, 1, l + N_{acc}(i,t) + 1, t+1$
5: **end for**
/* Phase 2: Meta-draft the draft following the UCB bandit until target sequence length B */
6: **while** $l < B$ **do**
7: $a_t \leftarrow \arg\max_{i \in [K]} \hat{\mu}_{i,t} + \beta\sqrt{\frac{2\ln t}{n_i}}$
8: Do one round of SD with drafter a_t and obtain $N_{acc}(a_t,t)$, $r_{a_t,t}$ (by eq. 1)
9: $\hat{\mu}_{i,t} \leftarrow \frac{1}{n_i(t)} \sum_{s=t-\tau+1}^{t} r_{a_s,s} \mathbb{I}[a_s = i] \; \forall i \in [K]$
10: $n_i(t) \leftarrow \sum_{s=t-\tau+1}^{t} \mathbb{I}[a_s = i] \; \forall i \in [K]$
11: $l, t \leftarrow l + N_{acc}(a_t,t) + 1, t+1$
12: **end while**

where, as before, B is the number of total tokens we have to generate, $\mu_{i,t}$ is the mean reward of choosing drafter i in t-th round, and $\tau(B)$ is the total round. However, the regret upper bound on eq. 28 does not always guarantee the performance of the SD as we discussed in Section 3.1. Instead, we can use our original regret objective using stopping time Definition 2 without any modification.

Here, we introduce two types of algorithms within our MetaSD framework: Discounted-UCB (D-UCB) algorithm [42] (Algorithm 6) and Sliding-window UCB [31] (Algorithm 7). Discounted UCB-SD estimates mean reward by computing the mean of discounted cumulative rewards as shown in the line 9 of Algorithm 6. By assigning less weight to the past observations, the algorithm finds a balance between accumulating knowledge and adapting to the changing environment. Similarly, sliding-window UCB utilizes a fixed-length window to calculate mean reward as demonstrated in the line 9-10 of Algorithm 7. By focusing only on recent information, it is also expected to achieve a balance with careful choose of the window size τ. [31].

One interesting point is that in the non-stationary MetaSD problem, the definition of non-stationarity L does not fit naturally into our problem. The reason behind this is that under non-stationary context generations, number of distribution changes happen at the token level, not the round level. This can disrupt existing regret analysis because a single round might involve multiple reward distribution

Algorithm 8: MetaSD-EXP3 [7]

INPUT Drafter pool $[K]$, initial prompt sequence $x^{1:l}$, target sequence length B, $\gamma \in (0, 1]$

$t \leftarrow 0$, $w_t(i) \leftarrow 1$ for $i = 1, \ldots, K$

while $l < B$ **do**

$$p_t(i) = (1-\gamma)\frac{w_t(i)}{\sum_{i=1}^{K} w_t(i)} + \frac{\gamma}{K} \quad i = 1, \ldots, K.$$

Draw a_t randomly according to the probabilities $p_t(1), \ldots, p_t(K)$.

Do one round of SD with drafter a_t and obtain $N_{acc}(a_t, t)$, $r_{a_t,t}$ (by eq. 1)

for $j = 1, \ldots, K$ **do**

$$\hat{r}_{j,t} = \begin{cases} r_{j,t}/p_t(j) & \text{if } j = a_t \\ 0 & \text{otherwise,} \end{cases}$$

$$w_{t+1}(j) = w_t(j) \exp\left(\frac{\gamma \cdot \hat{r}_{j,t}}{K}\right)$$

end for

$l, t \leftarrow l + N_{acc}(a_t, t) + 1, t + 1$

end while

changes (e.g., one round of speculative decoding could have two changing points). Whether above algorithms maintain optimal regret bounds in our regret definition in this non-stationary setting presents an interesting direction for future theoretical analysis.

H.3 Other possible scenarios

Adversarial environment EXP3 [7] is designed to handle adversarial changes of reward distributions by continuously updating its estimates of the arm rewards and adjusting its exploration strategy accordingly. It achieves this by maintaining a probability distribution over the arms and exponentially weighting the rewards based on their recent performance. By incorporating EXP3 into our framework (Algorithm 8), we can enable the system to adapt to evolving reward distributions and dynamically select the optimal drafter even in adversarial environments. We utilize this algorithm as a baseline in our experiments.

I Further discussion

I.1 Is scaling up drafter size always better?

While increasing the drafter size might seem like a straightforward path to improved performance, it can be less efficient than our MetaSD approach, especially considering memory bandwidth constraints. Larger models demand more memory for storing weights and activations, increasing data movement between memory and processing units. This can become a bottleneck, particularly in high-performance computing where memory bandwidth is often a limiting factor. It is also discussed in [74] in SD scenarios. Moreover, this phenomenon is well-illustrated by the roofline model, which highlights the trade-off between computational intensity and memory bandwidth [15]. As model size increases, computational intensity might improve, but the memory bandwidth demands can quickly limit overall speedup.

In contrast, MetaSD utilizes multiple smaller drafters with lower individual memory requirements. By efficiently switching between these drafters, MetaSD can achieve comparable or superior performance to a single large drafter while mitigating the memory bandwidth bottleneck. This is because, despite having multiple drafters, MetaSD only utilizes one drafter for computation at any given time. Thus, the memory bandwidth requirement does not scale with the combined size of all drafters, but rather with the size of the individual drafter being used. Provided sufficient GPU DRAM, this approach does not have any bottleneck compared to the single drafter SD. Furthermore, MetaSD offers the flexibility to incorporate diverse drafters with specialized capabilities. This specialization can

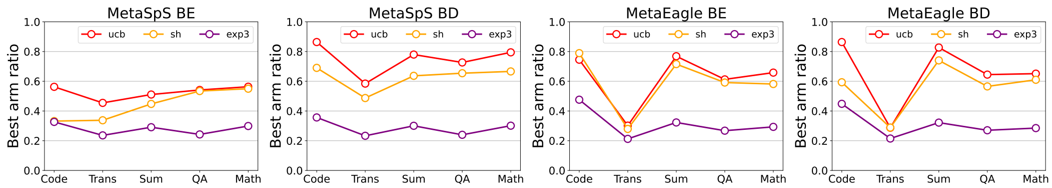


Figure 5: Best arm ratio over rounds for various MAB-SD configurations. (Left) MetaSpS (black-box SD) with BE and BD rewards. (Right) MetaEagle (white-box SD) with BE and BD rewards.

be more effective than simply increasing the size of a single general-purpose drafter, particularly for tasks demanding domain-specific knowledge.

I.2 Regret upper bound for MetaSD-UCB

Theorem 2 provides a regret upper bound for MetaSD-UCB, demonstrating that the number of rounds required to identify the optimal drafter is inversely proportional to the predefined draft length N_{max}. This aligns with the intuition that longer drafts provide more information about the relative performance of each drafter, leading to faster convergence towards the optimal choice. The logarithmic dependence on the budget B further highlights the efficiency of MetaSD-UCB in minimizing regret. These theoretical guarantees are supported by our empirical observations, where MetaSD-UCB consistently demonstrates strong performance and rapid convergence towards the best-performing drafter.

I.3 Memory bandwidth bound

A potential concern with our MetaSD framework is the increased memory bandwidth requirement due to loading multiple drafter models. However, our approach incurs minimal memory overhead. By storing all drafter weights in GPU DRAM, we avoid frequent accesses to slower system memory, which are a primary bottleneck for LLMs. For instance, with a 7B target LLM and float16 precision, our MetaEagle framework utilizes at most 19GB of GPU DRAM during generation, compared to 17GB for a single Eagle drafter. This represents only a small increase in memory usage, and importantly, it does not increase the memory bandwidth requirement during inference since only one drafter is active at a time.

J Further experiment

Best arm ratio To further analyze the behavior of MetaSD, we examine the best arm ratio, which represents the frequency of selecting the optimal drafter for a given task. Figure 5 illustrates how this ratio evolves over speculative decoding rounds, comparing different reward types (BE and BD) and bandit algorithms (SH, EXP3, UCB) for both MetaSpS (black-box SD) and MetaEagle (white-box SD). Across all configurations, UCB consistently identifies the best arm more rapidly than other bandit algorithms. This trend is particularly pronounced in the MetaSpS setting. Additionally, the BD reward generally leads to a higher best arm ratio compared to BE, suggesting that BD provides a more informative signal for drafter selection. This observation aligns with our earlier hypothesis that BD better captures the underlying dynamics of SD. Overall, the combination of UCB with the BD reward exhibits the most rapid convergence towards the optimal drafter.

Temperature sampling We investigate the impact of temperature sampling on MetaSpS performance. Table 8 presents the speedup ratios achieved with temperature sampling with temperature 0.7 on an NVIDIA A6000 GPU. Consistent with the trends observed in our main experiments with greedy decoding, MetaSD continues to achieve competitive speedup.

Table 8: Speedup ratio with temperature sampling as temperature is set to 0.7 over a NVIDIA A6000 GPU.

Dataset	SpS with specialized drafters					Bandit
	Drafter1	**Drafter2**	**Drafter3**	**Drafter4**	**Drafter5**	**UCB**
Code	**2.250**	1.215	1.379	1.532	1.513	1.896
Trans	1.086	**1.886**	1.096	1.130	1.078	1.431
Sum	1.461	1.165	**1.874**	1.463	1.353	1.744
QA	1.316	1.193	1.324	**1.776**	1.272	1.534
Math	1.450	1.258	1.355	1.616	**2.379**	2.046

Long-context De $\rightarrow$En translation While our results in Table 3 and Table 5 have the relatively less effectiveness of MetaSpS on the WMT16 De$\rightarrow$En translation task than other tasks, it is worth noting that this dataset primarily consists of relatively short sentences with an average length of fewer than 100 tokens. To assess the performance of our framework in a more challenging long-context scenario, we evaluate it on a new De$\rightarrow$En translation dataset with an average context length of 500 tokens generated by GPT-4o. As shown in Table 9, MetaSpS-UCB achieves a speedup ratio of 2.031 on this long-context dataset, approaching the performance of the optimal drafter (Drafter3).

Table 9: Speedup ratio on long-context De$\rightarrow$En translation with the same settings in Table 5.

Dataset	Drafter1	Drafter2	Drafter3	Drafter4	Drafter5	UCB
Long De$\rightarrow$En	1.238	1.316	2.044	0.970	1.187	2.031

Table 7: Mathematical terms and notations in our work.

Notation	Descriptions
K	Number of drafters
$[K]$	For a given integer K, denotes the set $\{1, ..., K\}$
i	Drafter index $i \in [K]$
α_i	True mean of acceptance rate when using drafter i
$i^\star$	Drafter index with the highest α_i
t	Number of current round
B	Total number of tokens to generate
$l(t)$	Number of input tokens at round t
$x^{1:l}$	Token sequence of first l tokens
$\mathcal{M}_q$	Target model
$\mathcal{M}_{q_i}$	The i-th drafter
p^l	Probability distribution of target model output given token sequence $x^{1:l}$
q_i^l	Probability distribution of output of drafter i given token sequence $x^{1:l}$
$N_{acc}(i,t)$	Number of accepted tokens using drafter i in round t
N_{max}	Number of candidate tokens
r	Arbitrary reward distribution with bounded support [0,1]
$r_{i,t}$	General reward feedback using drafter i in round t
$r_{i,t}^{BE}$	BE reward using drafter i in round t
$r_{i,t}^{BD}$	BD reward using drafter i in round t
$n_i(t)$	Number of selecting drafter i until round t
a_t	Index of selected drafter in round t
β	The exploration strength hyperparamter in UCB
γ	The exploration hyperparameter used in EXP3
μ_i	Expectation of the reward distribution of drafter i
π	Bandit policy (algorithm)
$\tau(\pi, B)$	Stopping time for the policy π with given total number of tokens B
λ	Switching cost constant factor
Δ_i	Suboptimality gap for the arbitrary reward distribution r: $\mu_i^\star - \mu_i$
$\Delta(\alpha_i)$	Suboptimality gap for the BD reward: $\alpha_i^\star - \alpha_i$
Δ_i^{BE}	Suboptimality gap for the BE reward
$R(r_i)$	Feedback signal for reward distribution when using drafter i (Theorem 1)
$d_{TV}(\cdot,\cdot)$	The total variation distance between probability measures
$\mathbb{I}$	Indicator function
$O(\cdot)$	Big O notation